TECHNICAL UNIVERSITY - SOFIA

FACULTY OF TELECOMMUNICATIONS

Department of Radio Communications and Video Technologies

RESEARCH PROJECT

BACHELOR OF TELECOMUNICATIONS DEGREE

TITLE: IMAGE CONTRAST ENHANCEMENT METHODS

SUPERVISOR: Assoc. Prof. Antoaneta Popova **STUDENT: Cristian Ordoyo Casado**

SOFIA 2010

0. Index

1. Analysis of contrast enhancement methods

1.1. Contrast definition

[1]Contrast is the difference in visual properties that makes an object (or its representation in an image) distinguishable from other objects and the background. In visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view. In other words, it is the different between the darker and the lighter pixel of the image, if it is big the image will have high contrast and in the other case the image will have low contrast.

Figure 1.1.1 On the left half low contrast, and on the right half high contrast image.

1.2. Contrast enhancement methods

[1][2][3][5]The principal objective of enhancement is to process an image so that the result is more suitable than the original image for a specific application. The word specific is important, because it establishes at the outset that the techniques are oriented to the problem. Thus, for example, a method that is quite useful for enhancing X-ray images may not necessarily be the best approach for enhancing pictures of Mars transmitted by a space probe. Regardless of the method used, however, image enhancement is one of the most interesting and visually appealing areas of image processing. Image enhancement approaches fall into two broad categories: spatial domain methods and frequency domain methods. The term spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. Frequency domain processing techniques are based on modifying the Fourier transform of an image. I use the spatial domain. Enhancement techniques based on various combinations of methods from these two categories are not unusual. There is no general theory of image enhancement. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular method works. Visual evaluation of image quality is a highly subjective process, thus making the definition of a “good image” an elusive standard by which to compare algorithm performance. When the problem is one of processing images for machine perception, the evaluation task is somewhat easier. For example, in dealing with a character recognition application, and leaving aside other issues such as computational requirements, the best image processing method would be the one yielding the best machine recognition results. However, even in situations when a clear-cut criterion of performance can be imposed on the

problem, a certain amount of trial and error usually is required before a particular image enhancement approach is selected.

1.3. Spatial domain transformations

[1][2][6][7][9][10][12]The term spatial domain refers to the aggregate of pixels composing an image. Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression

$$g(x,y) = T[f(x,y)] \qquad \textit{(Eq.1.3.1)}$$

where f(x, y) is the input image, g(x, y) is the processed image, and T is an operator on f, defined over some neighborhood of (x, y). In addition, T can operate on a set of input images, such as performing the pixel-by-pixel sum of K images for noise reduction. The principal approach in defining a neighborhood about a point (x, y) is to use a square or rectangular subimage area centered at (x, y). The center of the subimage is moved from pixel to pixel starting at the top left corner. The operator T is applied at each location (x, y) to yield the output, g, at that location. The process utilizes only the pixels in the area of the image spanned by the neighborhood. Although other neighborhood shapes, such as approximations to a circle, sometimes are used, square and rectangular arrays are by far the most predominant because of their ease of implementation. The simplest form of T is when the neighborhood is of size 1*1 (that is, a single pixel). In this case, g depends only on the value of f at (x, y), and T becomes a gray-level (also called an intensity or mapping) transformation function of the form

$$s = T[r] \qquad \textit{(Eq.1.3.2)}$$

where, for simplicity in notation, r and s are variables denoting, respectively, the gray level of f(x, y) and g(x, y) at any point (x, y).

1.4. Frequency domain transformations

[1][3][8][11]The French mathematician Jean Baptiste Joseph Fourier was born in 1768 in the town of Auxerre, France. The contribution for which he is most remembered was outlined in a memoir in 1807 and published in 1822 in his book, *La Théorie Analitique de la Chaleur* (The Analytic Theory of Heat). This book was translated into English 55 years later by Freeman 1878. Basically, Fourier's contribution in this particular field states that any function that periodically repeats itself can be expressed as the sum of sines and cosines of different frequencies, each multiplied by a different coefficient (we now call this sum a Fourier series). It does not matter how complicated the function is; as long as it is periodic and meets some mild mathematical conditions, it can be represented by a sum. This is now taken for granted, but at the time it first appeared it was a revolutionary concept to which it took mathematicians all over the world over a century to "adjust." At that time, regularity in functions was a mainstay of mathematical thinking. With this type of cultural mindset, the concept that complicated functions could be represented as a sum of simple sines and cosines was not at all intuitive (Figure 1.4.1), so it is not surprising that Fourier's ideas in this regard were met with skepticism.

Even functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and cosines multiplied by a weighing function. The formulation in this case is the Fourier transform, and its utility is even greater than the Fourier series in most practical problems. Both representations share the important characteristic that a function, expressed in either a Fourier series or transform, can be reconstructed (recovered) completely via an inverse process, with no loss of information. This is one of the most important characteristics of these representations because they allow us to work in the "Fourier domain" and then return to the original domain of the function without losing any information.

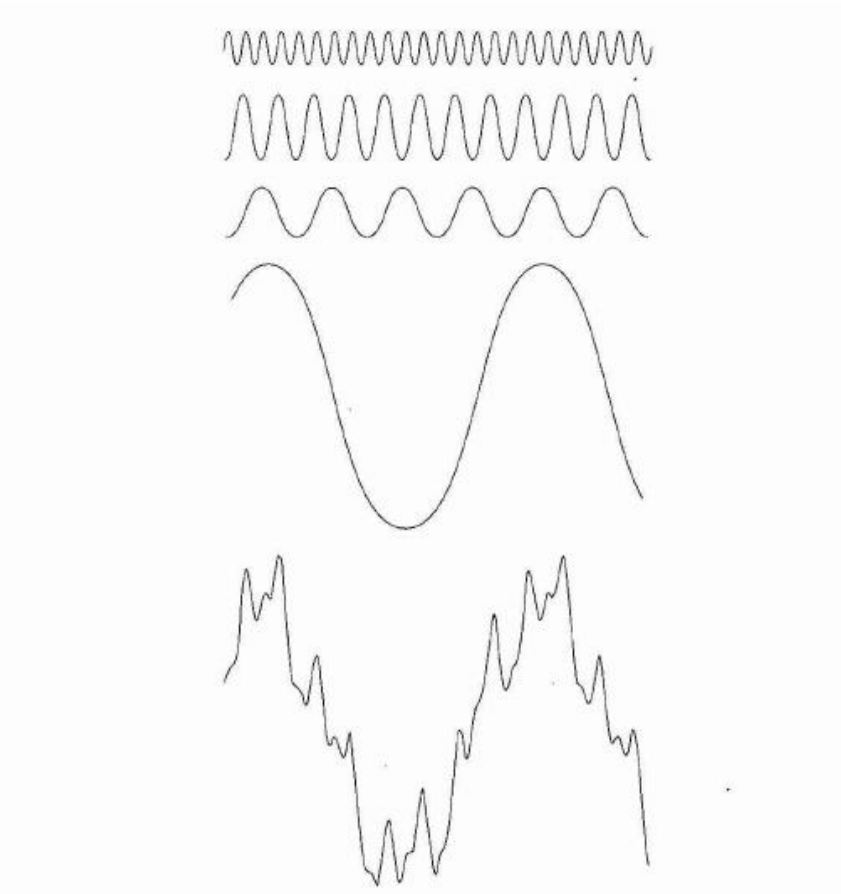


Figure 1.4.1 Plot of 4 sines and cosines and in the bottom its sum.

The advent of digital computation and the discovery of the Fast Fourier Transform (FFT) algorithm in the late 1950s revolutionized the field of signal processing. These two core technologies allowed for the first time practical processing and meaningful interpretation of a host of signals of exceptional human and industrial importance, from medical monitors and scanners to modern electronic communications.

We will be dealing only with functions (images) of finite duration, so the Fourier transform is the tool in which we are interested.

2. Mathematical methods explanation

2.1. Logarithmic transformation

The general for m is

$$s = c * \log(1 + r) \qquad \textit{(Eq.2.1.1)}$$

where s is the output value, r is the input value and c is a constant. This transformation maps a narrow range of low gray-level values in the input image into a wider range of output levels. The opposite is true of higher values of input levels. We would use this transformation to expand the values of dark pixels in an image while compressing the higher-level values. To expand the bright levels we would use the inverse logarithmic transformation.

2.2. Power-Law transformation

The general form is

$$s = c * r^{\gamma} \qquad \textit{(Eq.2.2.1)}$$

where c and γ are positive constants. As in the case of the logarithmic transformation, power-law curves with fractional values of γ

map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels. We see in Fig.2.2.2 that curves generated with values of $\gamma > 1$ have exactly the opposite effect as those generated with values of $\gamma < 1$. We also note that Eq.2.2.1 reduces to the identity transformation when $c = \gamma = 1$.

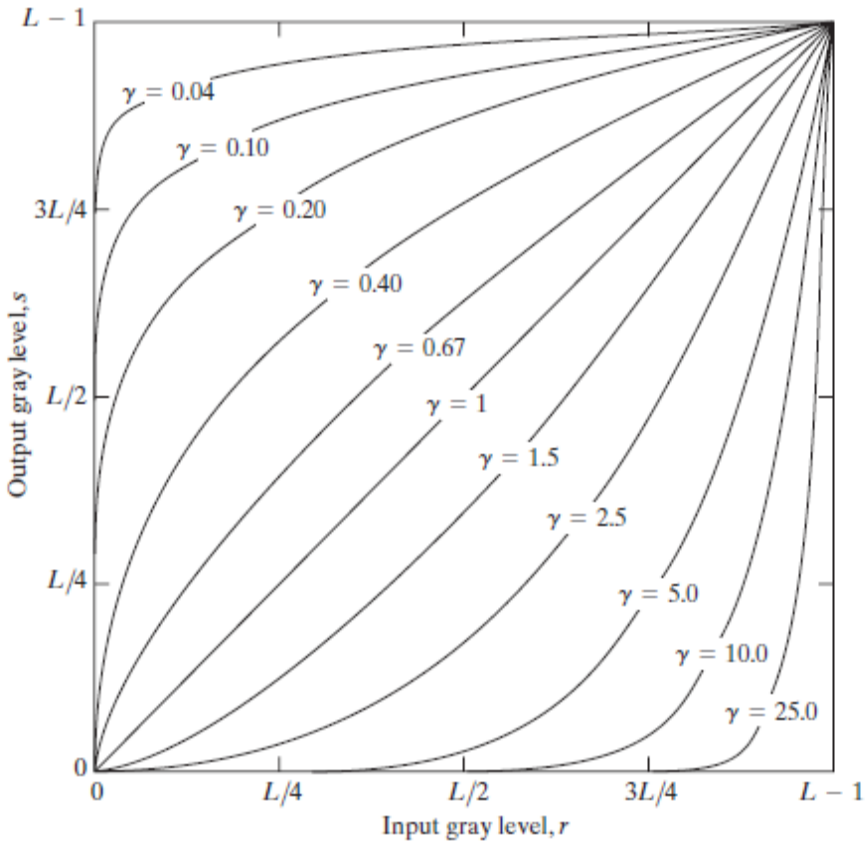


Figure 2.2.2 Plots of the equation for different values of γ, c= 1 in all cases.

A variety of devices used for image capture, printing, and display respond according to a power law. By convention, the exponent in the power-law equation is referred to as *gamma.* The process used to correct this power-law response phenomena is called *gamma correction.* For

example, cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function, with exponents varying from approximately 1.8 to 2.5. We see in Fig.2.2.3 that such display systems would tend to produce images that are darker than intended and gamma correction would correct it by preprocess the input.

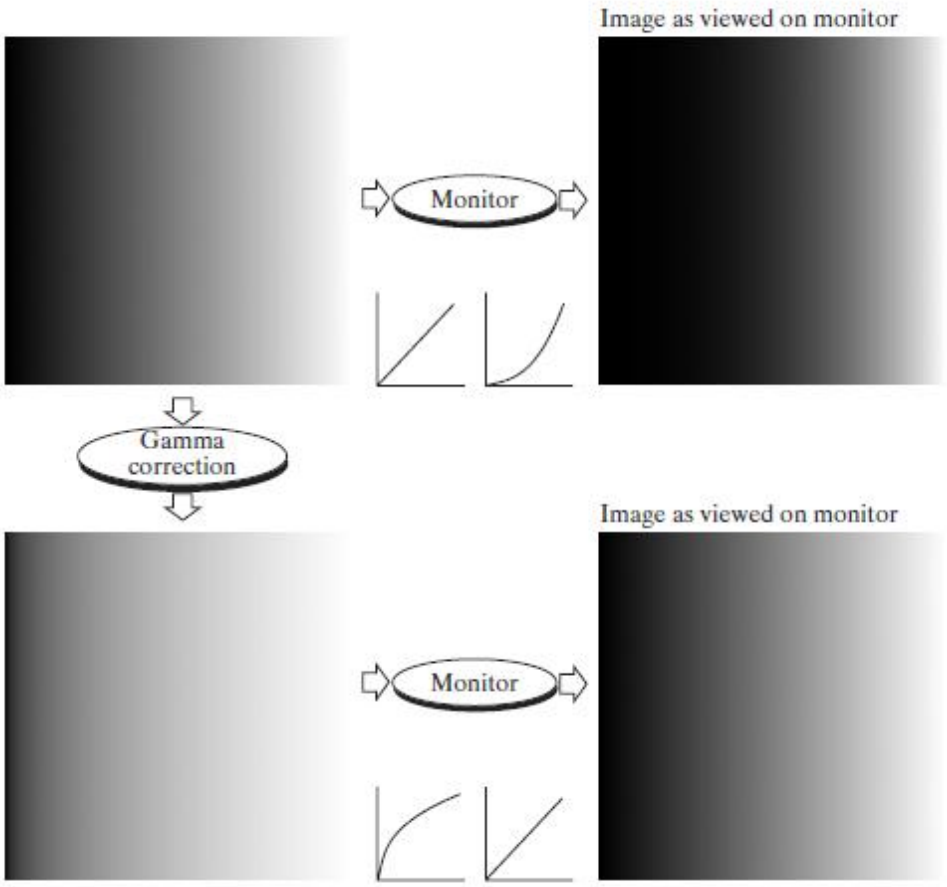


Figure 2.2.3 Gamma correction in CRT TV

2.3. Histogram equalization

The general form is

$$s_k = \frac{(L-1) * (r_k - r_{kmin})}{(r_{kmax} - r_{kmin})} \qquad \textit{(Eq.2.3.1)}$$

k = 0,1,2….L-1

where r and s are the input and output pixels of the image, L is the different values that can be the pixels, and r_{kmax} and r_{kmin} are the maximum and minimum gray values of the input image.

This method usually increases the global contrast of images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

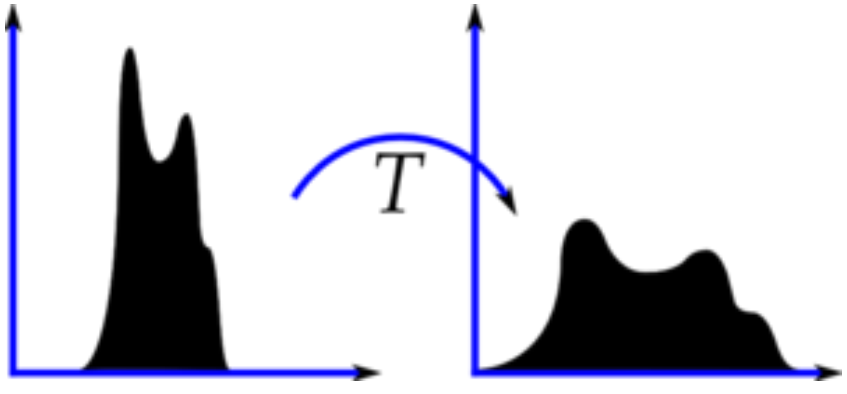


Figure 2.3.1 Histogram equalization

The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator. So in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.

2.4. Local enhancement using histogram statistics

This method is used to enhance details over small areas in an image. The procedure is to define a square or rectangular neighborhood and move the center of this area from pixel to pixel. At each location, the histogram of the points in the neighborhood is computed and either a histogram equalization or histogram specification transformation function is obtained. This function is finally used to map the gray level of the pixel centered in the neighborhood. The center of the neighborhood region is then moved to an adjacent pixel location and the procedure is repeated. Since only one new row or column of the neighborhood changes during a pixel-to-pixel translation of the region, updating the histogram obtained in the previous location with the new data introduced at each motion step is possible. This approach has obvious advantages over repeatedly computing the histogram over all pixels in the neighborhood region each time the region is moved one pixel location. Another approach used some times to reduce computation is to utilize non-overlapping regions, but this method usually produces an undesirable checkerboard effect.

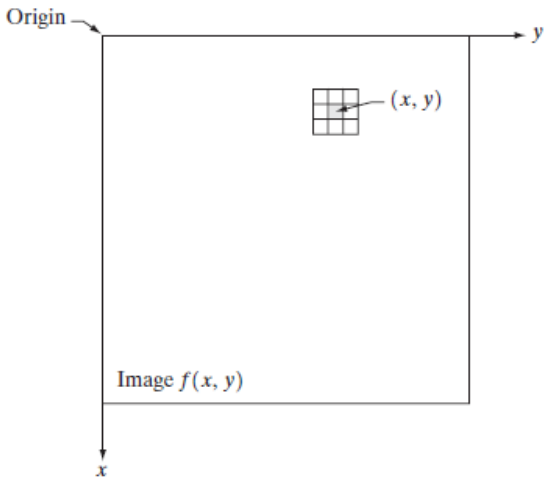


Figure 2.4.1 A 3x3 neighborhood about a point (x,y) in the image

We consider two uses of the mean and variance for enhancement purposes. The global mean and variance are measured over an entire image and are useful primarily for gross adjustments of overall intensity and contrast. A much more powerful use of these two measures is in local enhancement, where the local mean and variance are used as the basis for making changes that depend on image characteristics in a predefined region about each pixel in the image.

$$\boldsymbol{m} = \sum_{\boldsymbol{i=0}}^{\boldsymbol{L-1}} \boldsymbol{r}_{\mathrm{i}} * \boldsymbol{p}(\boldsymbol{r}_{\mathrm{i}}) \qquad \textit{(Eq.2.4.1)}$$

$$\mu_2 = \sum_{i=0}^{L-1} (r_{\mathrm{i}} - m)^2 * p(r_{\mathrm{i}}) \qquad \textit{(Eq.2.4.2)}$$

Let (x, y) be the coordinates of a pixel in an image, and let S_{xy} denote a neighborhood (subimage) of specified size, centered at (x, y).From (Eq.2.4.3) the mean value m_{Sxy} of the pixels in S_{xy} can be computed using the expression

$$m_{s_{xy}} = \sum_{(s,t)\in s_{xy}} r_{s,t} * p(r_{s,t}) \qquad \textit{Eq.2.4.3)}$$

where $r_{s,\,t}$ is the gray level at coordinates (s, t) in the neighborhood, and $p(r_{s,\,t})$ is the neighborhood normalized histogram component

corresponding to that value of gray level. Similarly, from (Eq.2.4.2), the gray-level variance of the pixels in region S_{xy} is given by

$$\sigma^2_{s_{xy}} = \sum_{(s,t)\in s_{xy}} (r_{s,t} - m_{s_{xy}})^2 * p(r_{s,t}) \qquad \textit{(Eq.2.4.4)}$$

The local mean is a measure of average gray level in neighborhood S_{xy}, and the variance (or standard deviation) is a measure of contrast in that neighborhood. An important aspect of image processing using the local mean and variance is the flexibility they afford in developing simple, yet powerful enhancement techniques based on statistical measures that have a close, predictable correspondence with image appearance.

2.5. Contrast stretching

One of the simplest piecewise linear functions is a contrast-stretching transformation. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture during image acquisition. The idea behind contrast stretching is to increase the dynamic range of the gray levels in the image being processed.

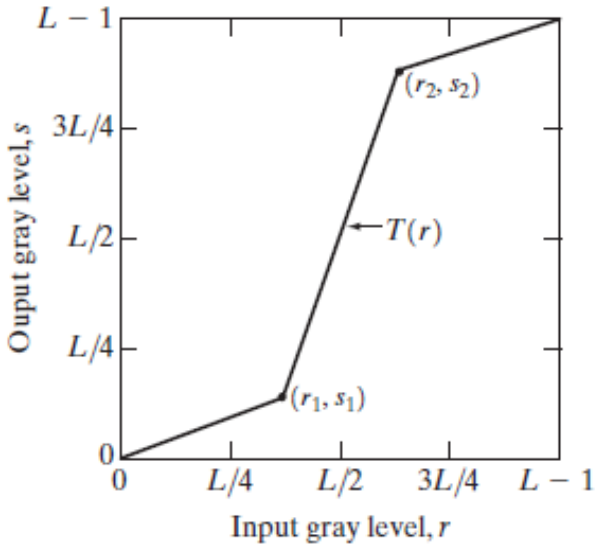


Figure 2.5.1 Contrast stretching explanation

The locations of points (r1, s1) and (r2, s2) control the shape of the transformation function. If $r_1=s_1$ and $r_2=s_2$, the transformation is a linear function that produces no changes in gray levels. If $r_1=r_2$, $s_1=0$ and $s_2=L-1$, the transformation becomes a thresholding function that creates a binary image. Intermediate values of (r_1, s_1) and (r_2, s_2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast. In general, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed so that the function is single

valued and monotonically increasing. This condition preserves the order of gray levels, thus preventing the creation of intensity artifacts in the processed image.

The general form is

$$s = \frac{1}{1+(m/r)^E} \qquad \textit{(Eq.2.5.2)}$$

where r are the input image values, s are the output image values, m is the thresholding value and E the slope.

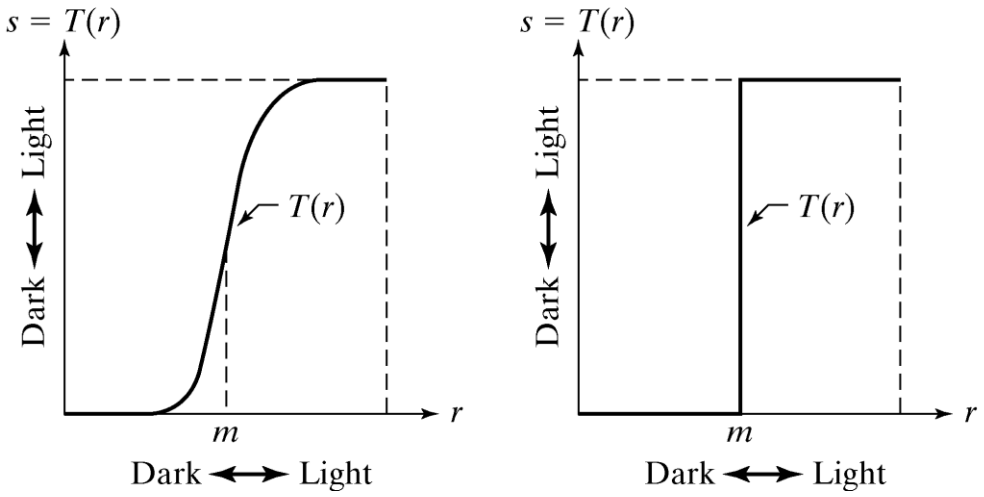


Figure 2.5.3 Effect of the slope

Figure 2.5.3 it show the effect of the variable E, if E = 1 the stretching became a threshold transformation, if E > 1 the transformation

its defined by the curve which is smoother when the E value is increase, and when E < 1 the transformation makes the negative and also stretching.

3. Programming Algorithms Simulation

In this chapter it is going to be show the flowchart of algorithms and also the implemented form in the simulation software MatLab.

3.1. Logarithmic transformation

3.1.1. Flowchart

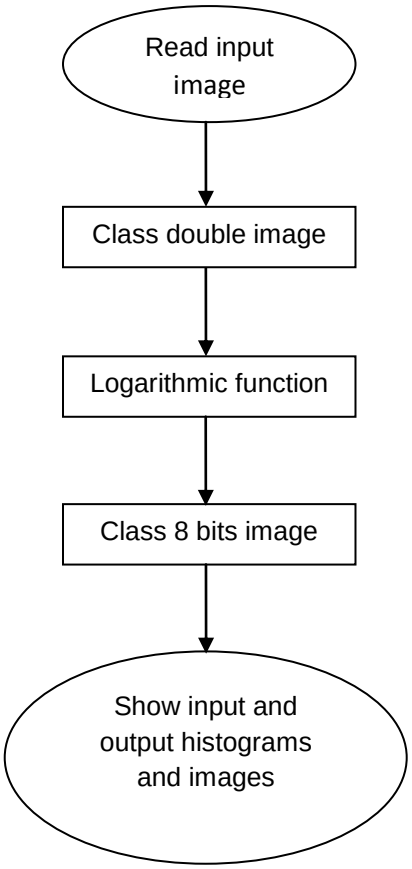


Figure 3.1.1.1 Logarithmic transformation flowchart

3.1.2. Algorithm

```
clear all, close all;

input_name='text39.jpg';
output_name='text39_enh.jpg';


input_image=imread(fullfile('infiles\',input_name));

input_image_process=double(input_image);

%logaritm transform
tic
output_image_process=log(1+input_image_process);
time=toc;
output_image=im2uint8(mat2gray(output_image_process));

input_hist=imhist(input_image);
output_hist=imhist(output_image);



subplot(2,2,1),imshow(input_image),title('Input image')
subplot(2,2,2),imshow(output_image),title('Output image')
subplot(2,2,3),plot(input_hist),title('Input histogram')

xlabel('Gray levels'),ylabel('Relative frecuency')
set(gca, 'xlim', [0 255]);

subplot(2,2,4),plot(output_hist),title('Output histogram')
xlabel('Gray levels'),ylabel('Relative frecuency')
set(gca, 'xlim', [0 255]);



imwrite(output_image,fullfile('outfiles\',output_name));
```

3.2. Power-Law transformation

3.2.1. Flowchart

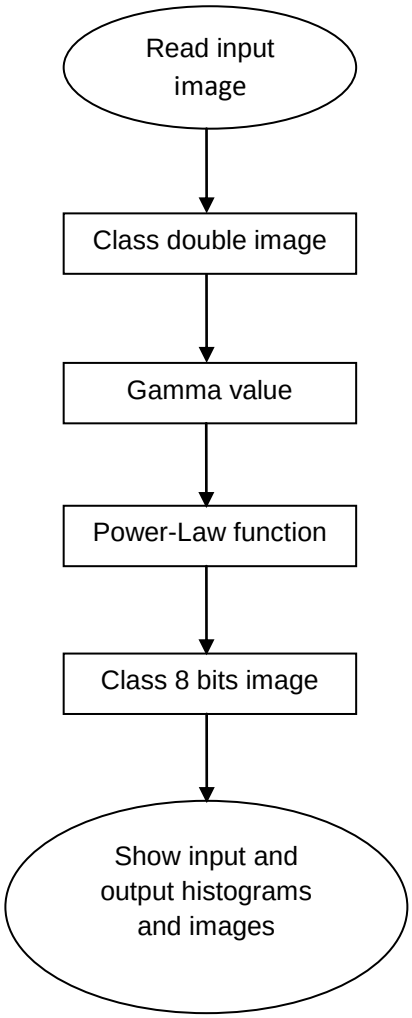


Figure 3.2.1.1 Power-Law transformation flowchart

3.2.2. Algorithm

```
clear all, close all;

input_name='text04.jpg';
output_name='text04_enh.jpg';

input_image=imread(fullfile('infiles\',input_name));

input_image_process=input_image;

%power transform gamma<1 increase the contrast in dark gamma>1
increase
%the contrast in whites

gamma=2;
tic
output_image_process=imadjust(input_image_process,[0 1],[0 1],gamma);
time=toc;
output_image=im2uint8(mat2gray(output_image_process));

input_hist=imhist(input_image);
output_hist=imhist(output_image);

subplot(2,2,1),imshow(input_image),title('Input image')
subplot(2,2,2),imshow(output_image),title('Output image')
subplot(2,2,3),plot(input_hist),title('Input histogram')

xlabel('Gray levels'),ylabel('Relative frecuency')
set(gca, 'xlim', [0 255]);

subplot(2,2,4),plot(output_hist),title('Output histogram')
xlabel('Gray levels'),ylabel('Relative frecuency')
set(gca, 'xlim', [0 255]);

imwrite(output_image,fullfile('outfiles\',output_name));
```

3.3. Histogram equalization

3.3.1. Flowchart

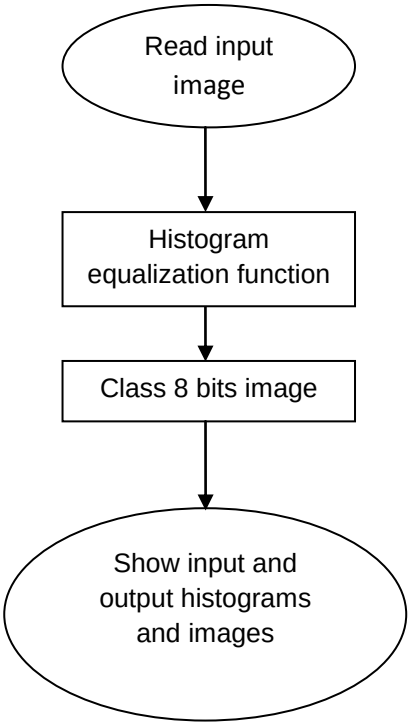


Figure 3.3.1.1 Histogram equalization flowchart

3.3.2. Algorithm

```
clear all, close all;

input_name='text62.jpg';
output_name='text62_enh.jpg';

input_image=imread(fullfile('infiles\',input_name));

input_image_process=input_image;

output_image_process=histeq(input_image_process);

output_image=im2uint8(mat2gray(output_image_process));

input_hist=imhist(input_image);
output_hist=imhist(output_image);



subplot(2,2,1),imshow(input_image),title('Input image')
subplot(2,2,2),imshow(output_image),title('Output image')
subplot(2,2,3),plot(input_hist),title('Input histogram')

xlabel('Gray levels')
ylabel('Relative frecuency')
set(gca, 'xlim', [0 255]);

subplot(2,2,4),plot(output_hist),title('Output histogram')
xlabel('Gray levels'),ylabel('Relative frecuency')

set(gca, 'xlim', [0 255]);



imwrite(output_image,fullfile('outfiles\',output_name));
```

3.4. Local enhancement using histogram statistics

3.4.1. Flowchart

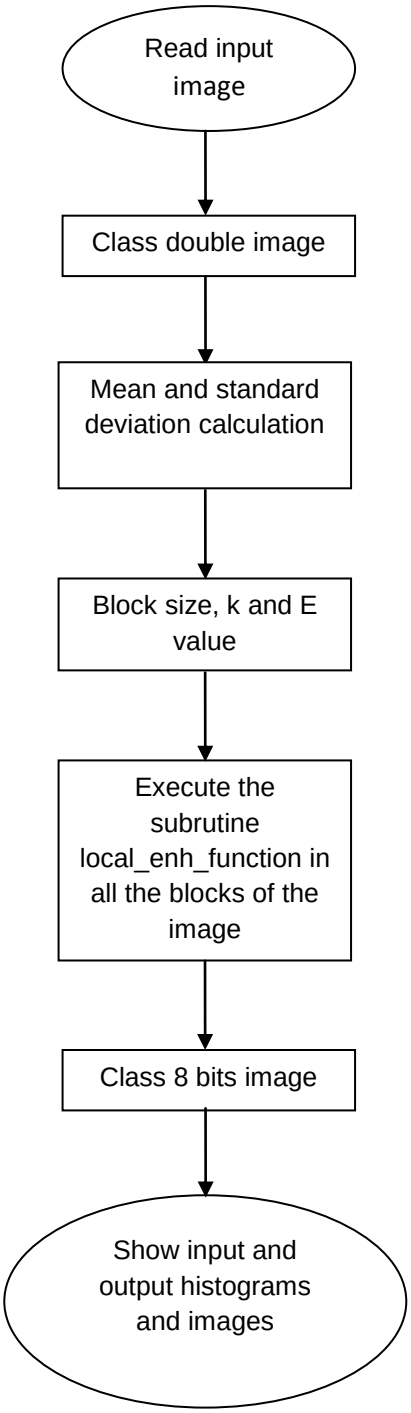


Figure 3.4.1.1 Local enhancement transformation, principal function flowchart

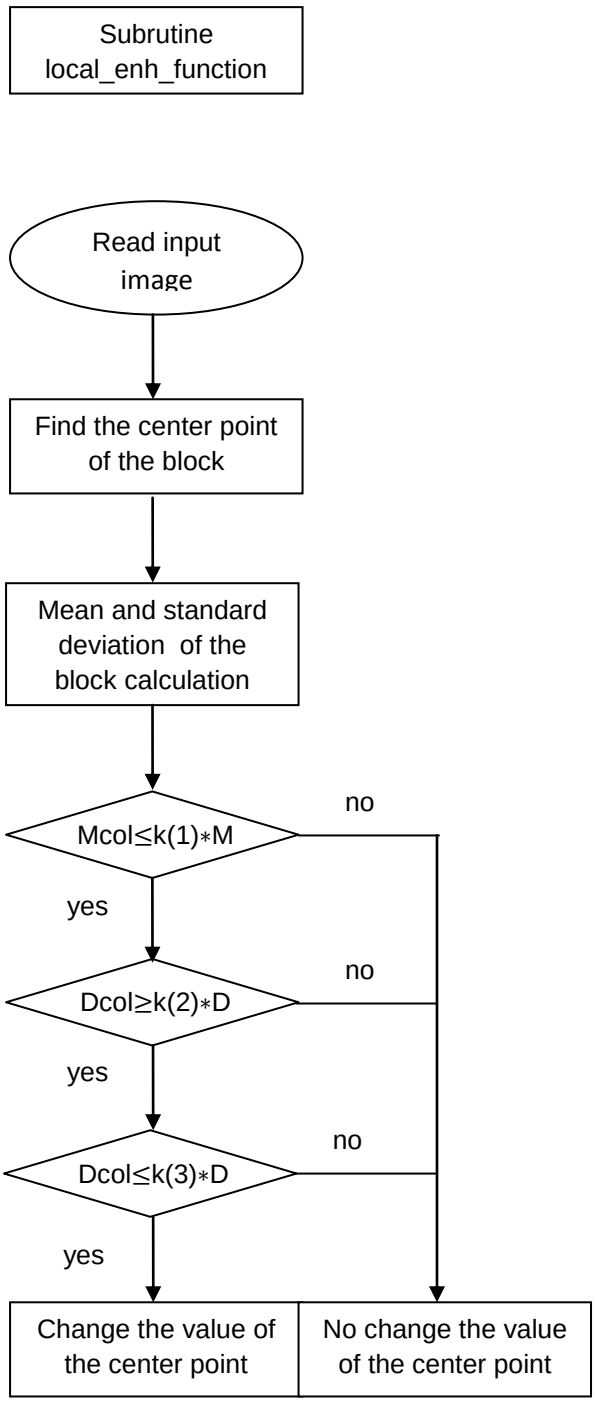


Figure 3.4.1.2 Local enhancement transformation, subrutine flowchart

3.4.2. Algorithm

```
clear all, close all;

input_name='test.jpg';
output_name='test_enh.jpg';

input_image=imread(fullfile('infiles\',input_name));


input_image_process=double(input_image);

M=mean2(input_image_process);
D=std2(input_image_process);

Bsize=[3 3];
k=[0.4 0.02 0.4];
E=4;

tic

output_image_process=colfilt(input_image_process,Bsize,'sliding',
@local_enh_function,M,D,E,k);

time=toc %display time-consuming


output_image=im2uint8(mat2gray(output_image_process));

subplot(2,2,1),imshow(input_image),title('Input image')
subplot(2,2,2),imshow(output_image),title('Output image')
subplot(2,2,3),plot(input_hist),title('Input histogram')

xlabel('Gray levels')
ylabel('Relative frecuency')
set(gca, 'xlim', [0 255]);

subplot(2,2,4),plot(output_hist),title('Output histogram')
xlabel('Gray levels'),ylabel('Relative frecuency')

set(gca, 'xlim', [0 255]);



imwrite(output_image,fullfile('outfiles\',output_name));
```

```
function g=local_enh_function(Icol,M,D,E,k)

Bcenter=floor((size(Icol,1)+1)/2);
g=Icol(Bcenter,:);


%Compute the local mean and variance.
Mcol=mean(Icol);
Dcol=std(Icol);

%Build the local response.
change=find((Mcol<=k(1)*M) & (Dcol>=k(2)*D) & (Dcol<=k(3)*D));
nochange=find((Mcol>k(1)*M) | (Dcol<k(2)*D) | (Dcol>k(3)*D));
g(change)=E*Icol(Bcenter,change);
%g(change)=255;g(nochange)=0;
```

3.5. Contrast stretching

3.5.1. Flowchart

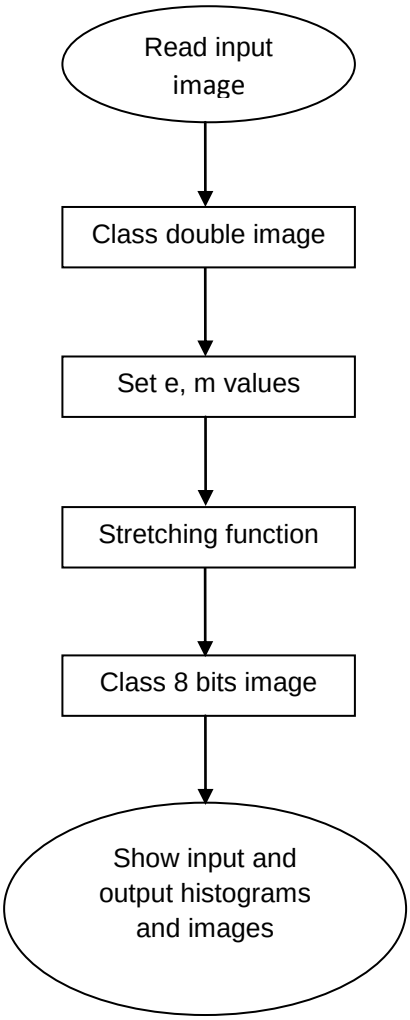


Figure 3.5.1.1 Contrast stretching flowchart

3.5.2. Algorithm

```
clear all, close all;

input_name='text07.jpg';
output_name='text07_enh.jpg';

input_image=imread(fullfile('infiles\',input_name));

input_image_process=double(input_image);

e=3;
m=80;

tic
output_image_process = 1 ./ (1 + (m./input_image_process).^e);
time=toc;

output_image=im2uint8(mat2gray(output_image_process));

input_hist=imhist(input_image);
output_hist=imhist(output_image);



subplot(2,2,1),imshow(input_image),title('Input image')
subplot(2,2,2),imshow(output_image),title('Output image')
subplot(2,2,3),plot(input_hist),title('Input histogram')

xlabel('Gray levels')
ylabel('Relative frecuency')
set(gca, 'xlim', [0 255]);

subplot(2,2,4),plot(output_hist),title('Output histogram')
xlabel('Gray levels'),ylabel('Relative frecuency')

set(gca, 'xlim', [0 255]);



imwrite(output_image,fullfile('outfiles\',output_name));
```

4. Experimental results

4.1. Logarithmic transformation

The logarithmic transformation gives more details in the darks areas making the image lighter and the light areas lost its details. The algorithm is very fast and the average time is 0.0392 seconds for a 240x320 image.

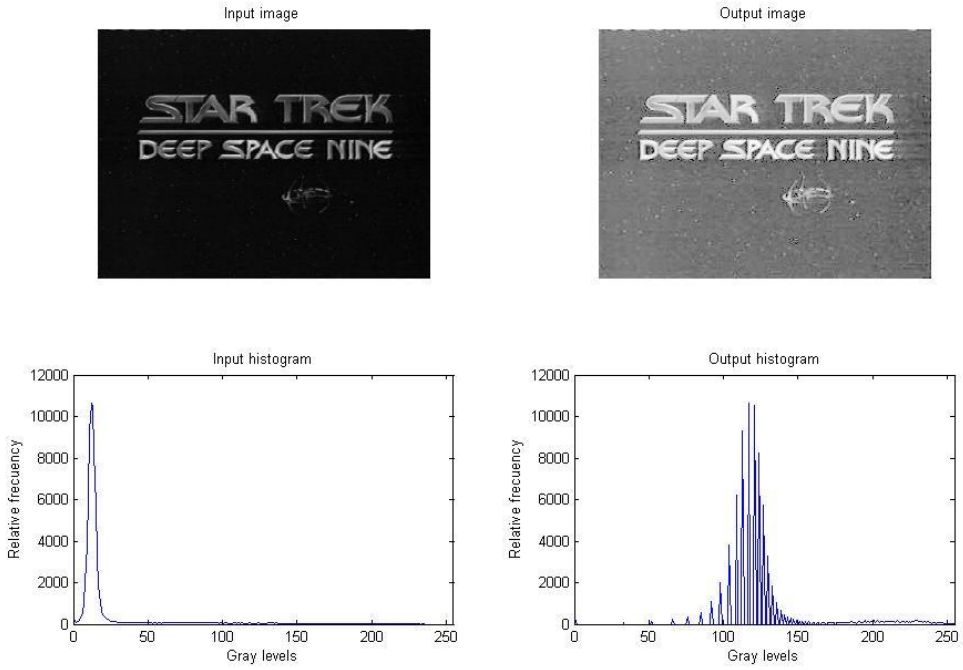


Figure 4.1.1 Logarithmic transformation, example 1.

The input image is almost black just the letters are in gray, also in the histogram all the pixels luminance values are near the 0 (black). After the logarithmic transform the output image it became lighter, in fact, an object behind the letter can be show better. The output histogram is more centered and more wide it has more gray levels than the first one.

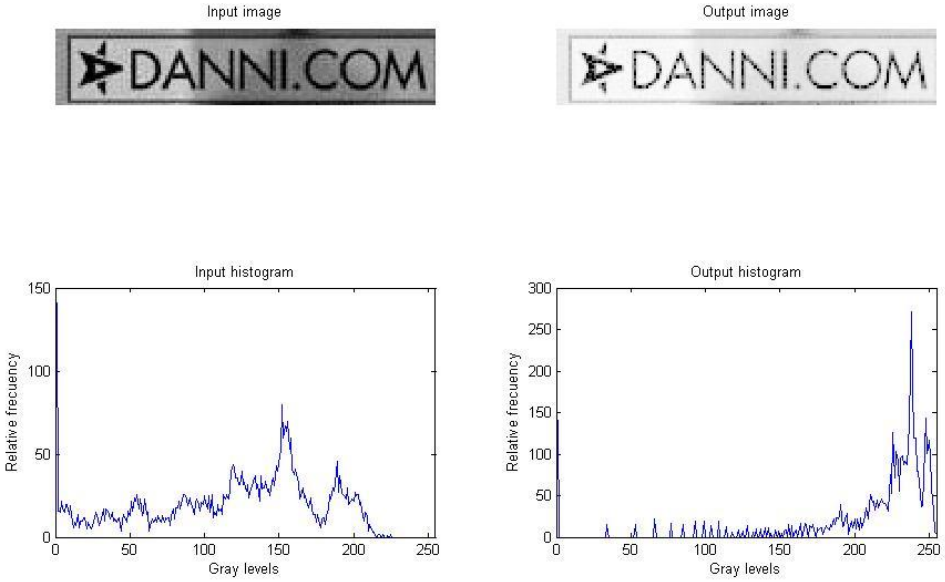


Figure 4.1.2 Logarithmic transformation, example 2.

This image has more levels of gray like it is show in its histogram and after the logarithm transformation the output histogram show that the darker pixels of the input image has been reduce and all the image seems lighter almost without values under lever 190.

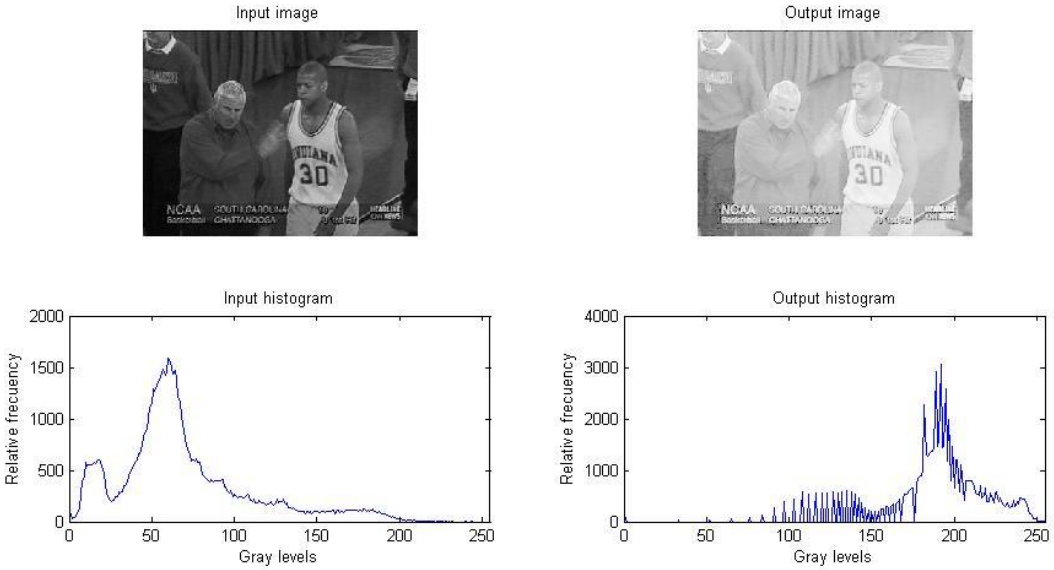


Figure 4.1.3 Logarithmic transformation, example 3.

Input image is a TV capture and most of the image is dark, after the transformation image became lighter and just have enhance the letters in the bottom. The contrast range do not change in a high form because the input image it is not dark enough.

4.2 Power-Law transformation

The power-law transformation have two ways to operate it depends of the gamma value. If gamma is lowest than 1 it is more or less like an logarithm transform but much better because it can be use different slopes to the function. Thus, the dark areas are enhanced and more detailed. And if gamma is highest than 1 the function does the inverse result. It is enhanced the light areas and makes the image darker. The algorithm average time is 0.1232 seconds for a 240x320 image. It is slower than the logarithm transform but gets better results.

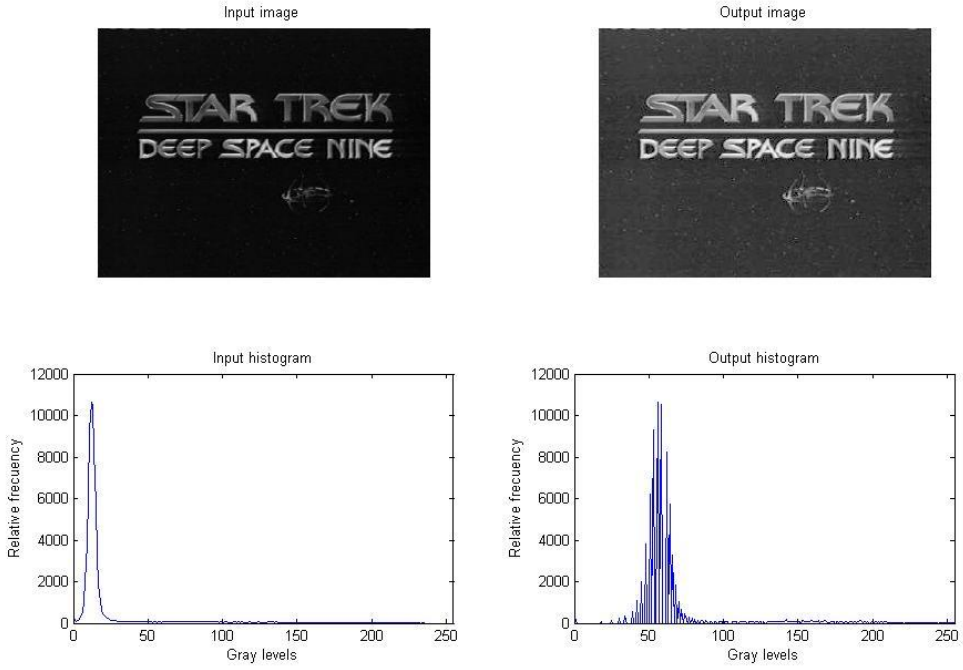


Figure 4.2.1 Power-law transformation, example 1.

The output image is lighter and also the dark areas get more details that in the input image. Its histograms show also this enhancement and

the gray values have been increase. Using different values of gamma get different results and using 0.5 it seems better result than using logarithm transformation.

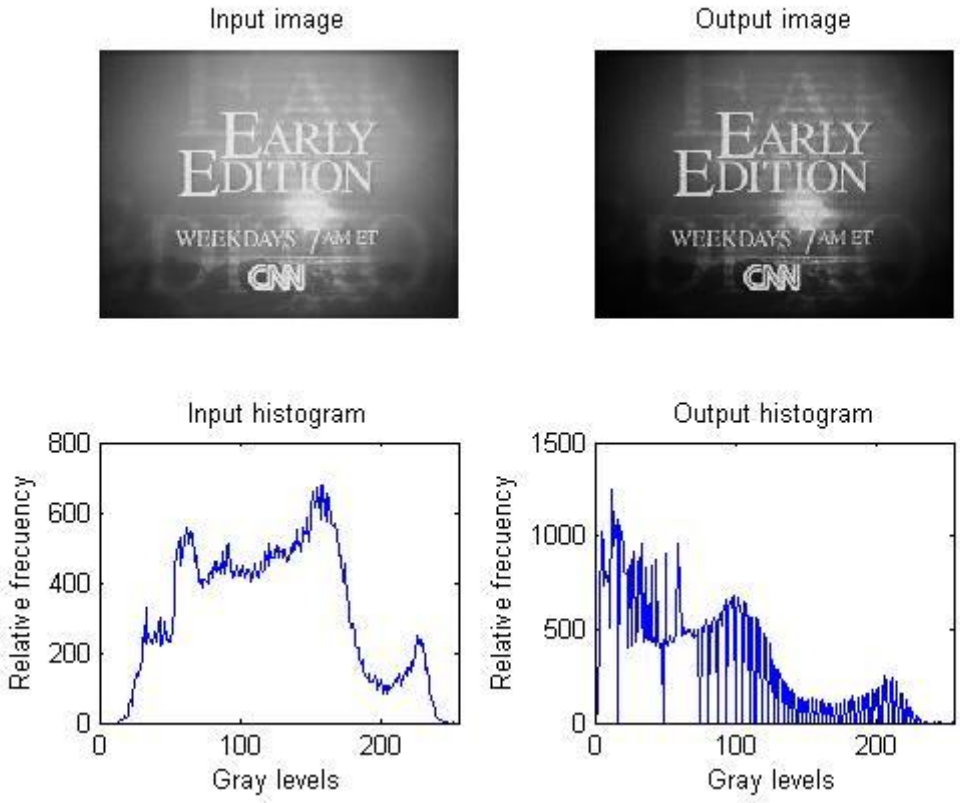


Figure 4.2.2 Power-Law transformation, example 2.

Using a gamma value higher than 1, in this case gamma is 2, the output image became darker and the light areas gets more detail, for example, the 7 of the picture has been enhanced and gets its edges with more detail. Thus, is is show in the histograms that the output image has increase the low gray levels (black) and decrease in the lighter grays.

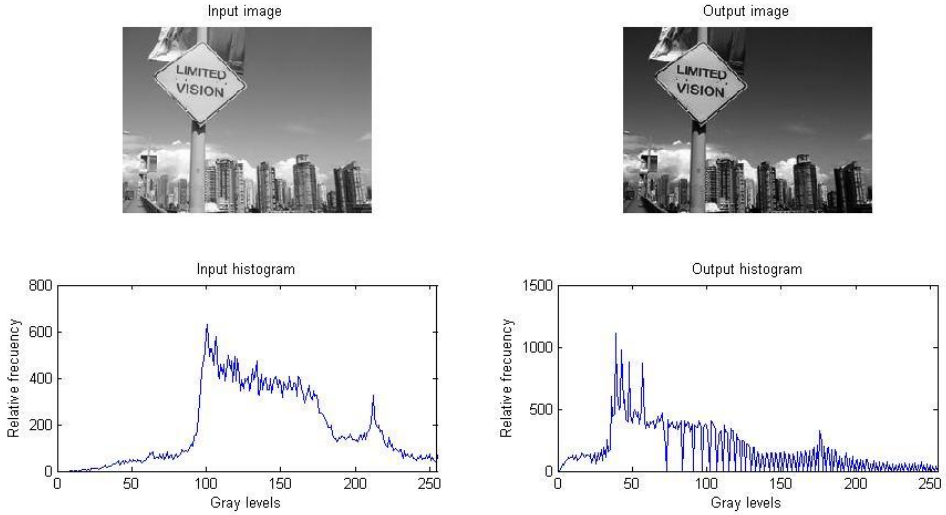


Figure 4.2.3 Power-Law transformation, example 3.

In example 3, also use gamma 2 because the objective it is enhance the light area of the image. In fact, the clouds of the background has a more realistic aspect after the transformation. The image became darker and the letter gets better contrast.

4.3 Histogram equalization

Histogram equalization makes the histogram to expand between all the range (0,255) and gets more smooth transitions between the pixels of the image. The algorithm average time is 0.1590 seconds for a 240x320 image. It is quite fast because we can process more than 6 images per second.

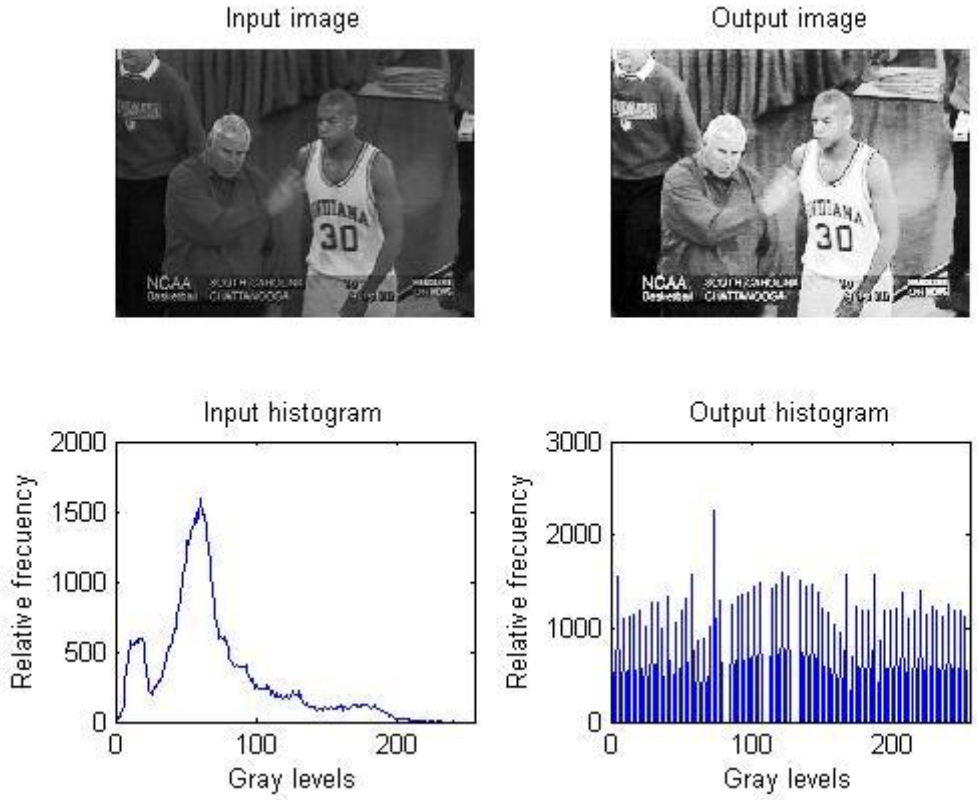


Figure 4.3.1 Histogram equalization, example 1.

This transformation affect to the histogram as it is show in the output it has pixels values in all the grayscale range. However in the input histogram the levels goes from 0 to 200, so the output image has increase its dynamic range in 50 values.

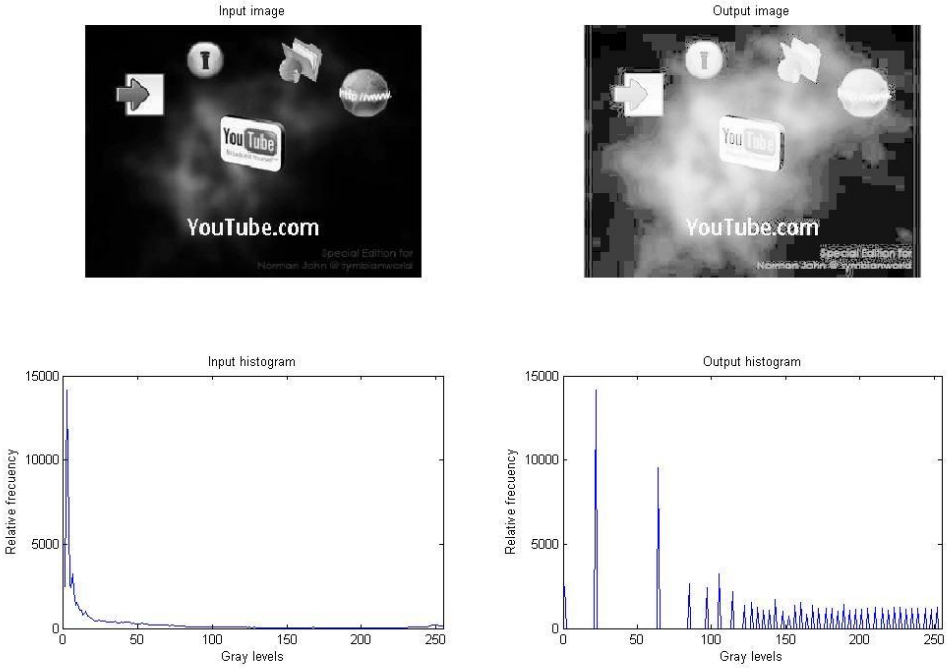


Figure 4.3.2 Histogram equalization, example 2.

Here it is most significative the variation of the histogram because in the beginning the histogram is almost black and after the equalization it have lots of medium level gray values hold also the black level but in less pixels.

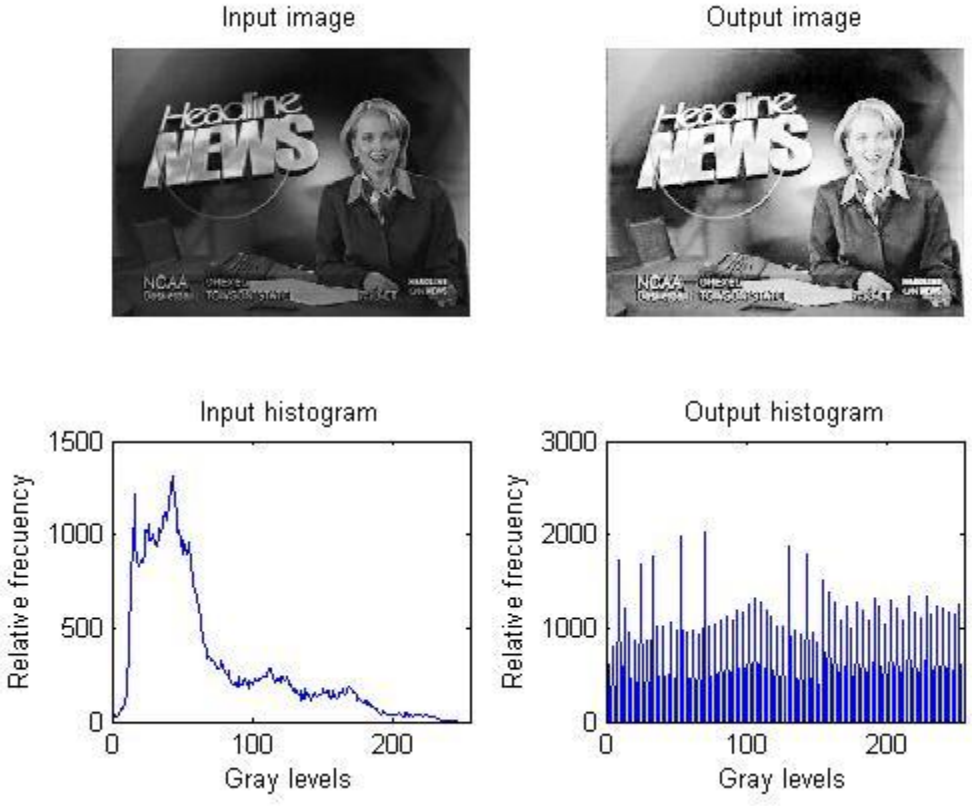


Figure 4.3.2 Histogram equalization, example 2.

As the input histogram show the image has low contrast and is dark in the beginning. After the transformation all gray levels takes similar number of pixels and makes lighter the image. Output image gets better contrast and it is easier read the letters in the bottom.

4.4 Local enhancement using histogram statistics

Local enhancement it is to improve the contrast on an image that has different partial areas, for example, an image that need to be improve in dark areas and also in the light areas, then it would be used this algorithm. The clue points of this method is to chose well the values of the relative dispersion to find the optimum pixels that it has to improve. The average time is almost half a second, 0.4208, but it is normal because it has to make the calculations with all subimages.

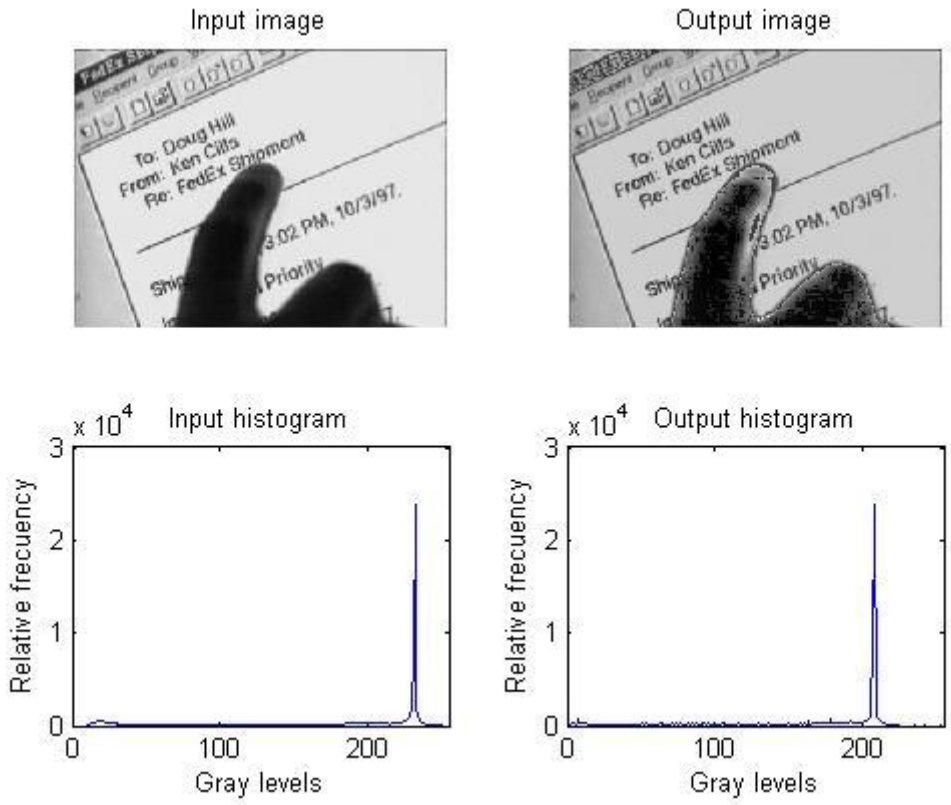


Figure 4.4.1 Local enhancement, example 1.

The input image has one white area and another one black that have low contrast. After the local enhancement transformation the output

image has more less the same contrast in the white area but has improve the black one contrast.

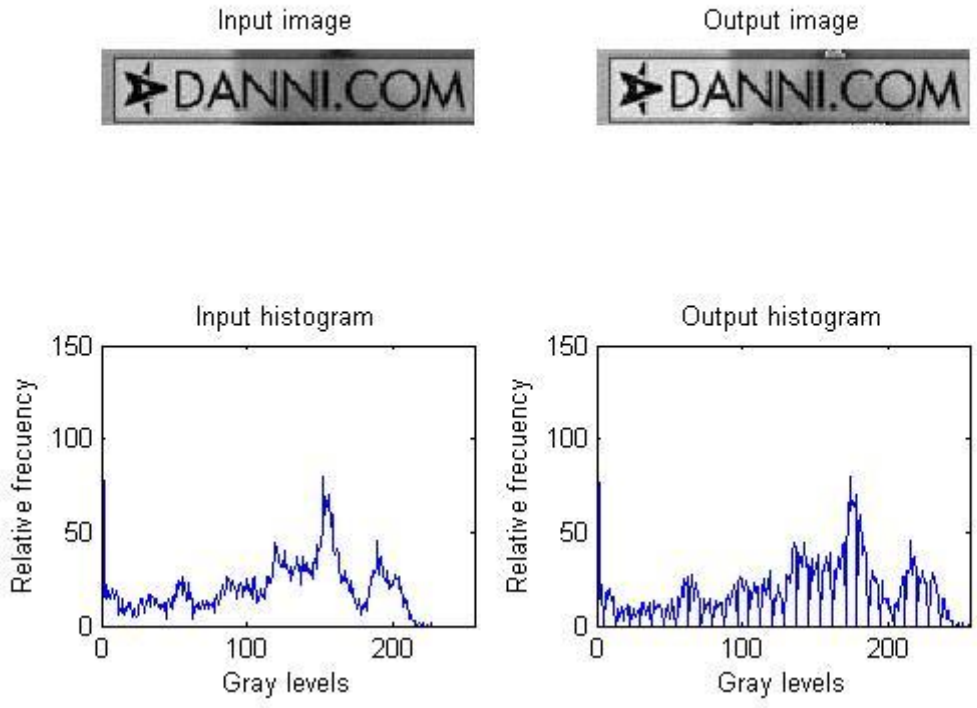


Figure 4.4.2 Local enhancement, example 2.

In this input image it is tried to enhance the black level (characters) because it has different backcourts. This local method use the neighbors levels to choose the correct gray value. The two images are similar but looking to the histograms it shows that the output has quite more light levels than the first one.

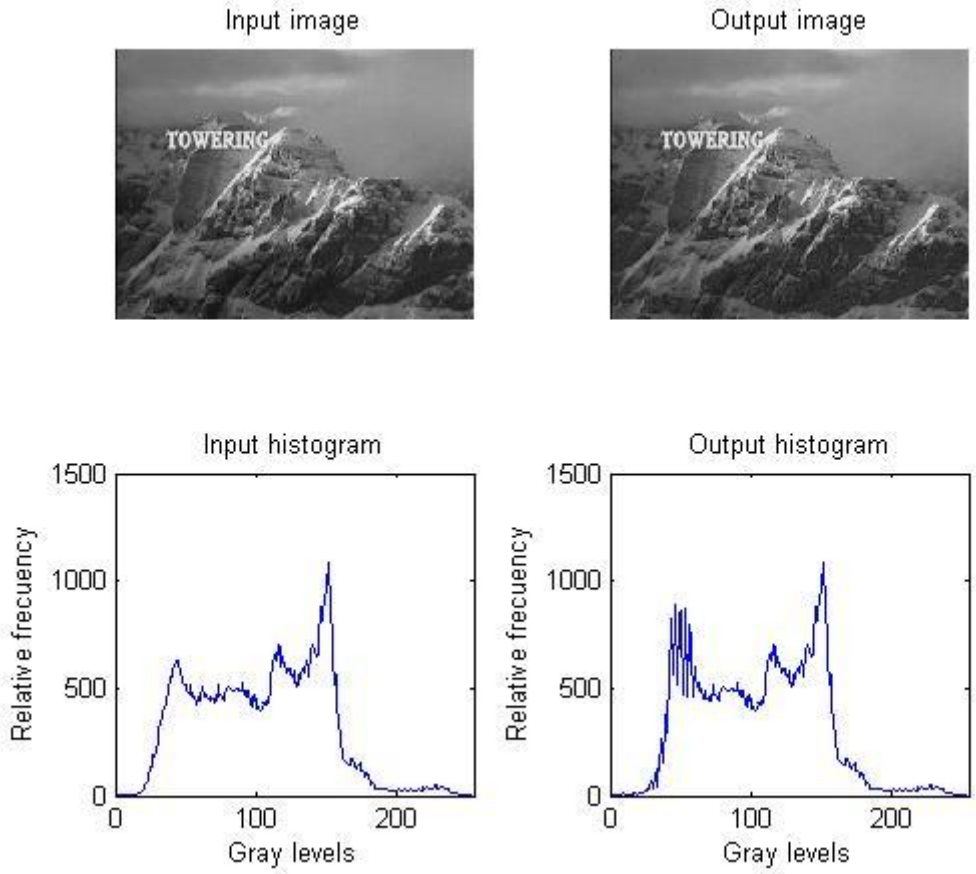


Figure 4.4.2 Local enhancement, example 2.

Looking the output histogram dark values has increase, and the bottom side of the image has improve the contrast in the dark area, not the whites.

4.5. Contrast stretching

Contrast stretching separate the image in two parts the black and the white one, on the m value, and the transition between these parts is a slope that could be more or less smooth in the depends on the e value. It is also a fast algorithm and the average time is 0.0631

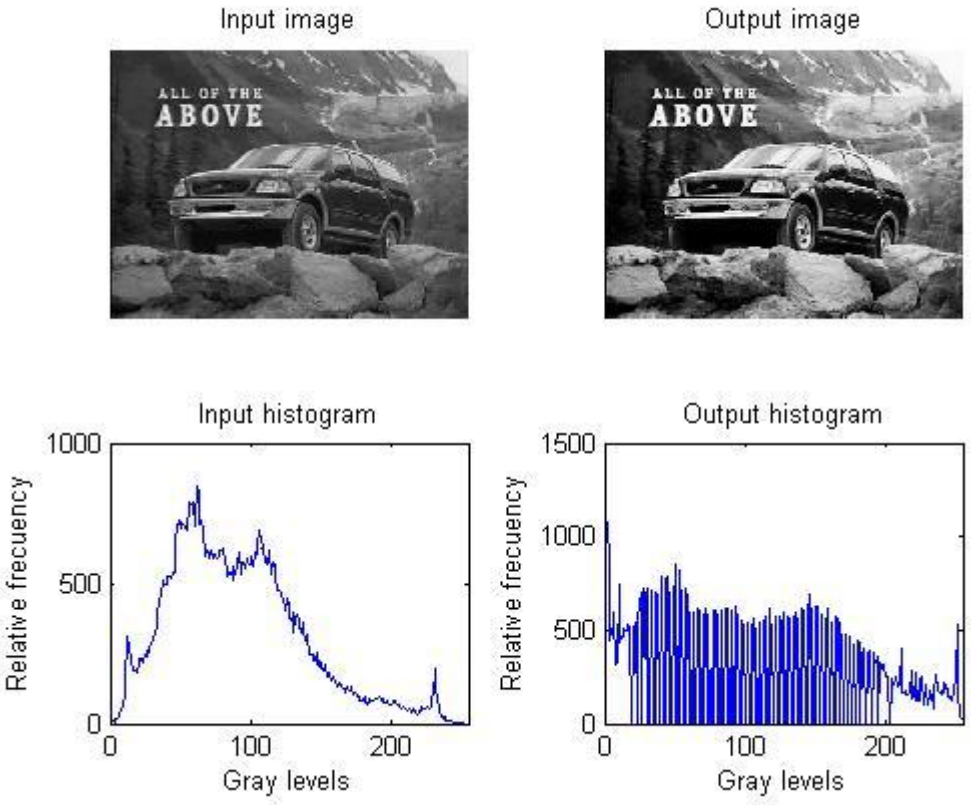


Figure 4.5.1 Contrast stretching, example 1.

The input image has a black corner and the rest of the image is different gray values. After the transformation it is darker in the dark areas and lighter in the light areas so it is increase the contrast. It has been used values for m and e of 100 and 3.

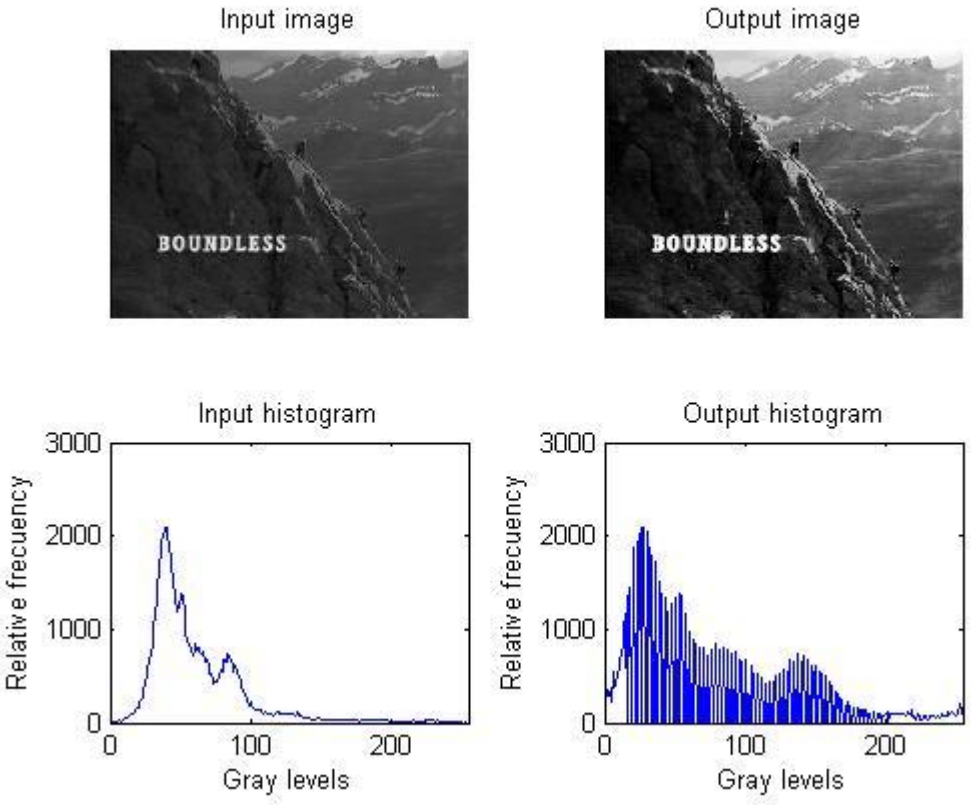


Figure 4.5.2 Contrast stretching, example 2.

The input image is so dark and it is difficult to see the alpinist but after the transformation with 80 for the m variable and 3 for the e variable the contrast has been enhanced and they are better show.

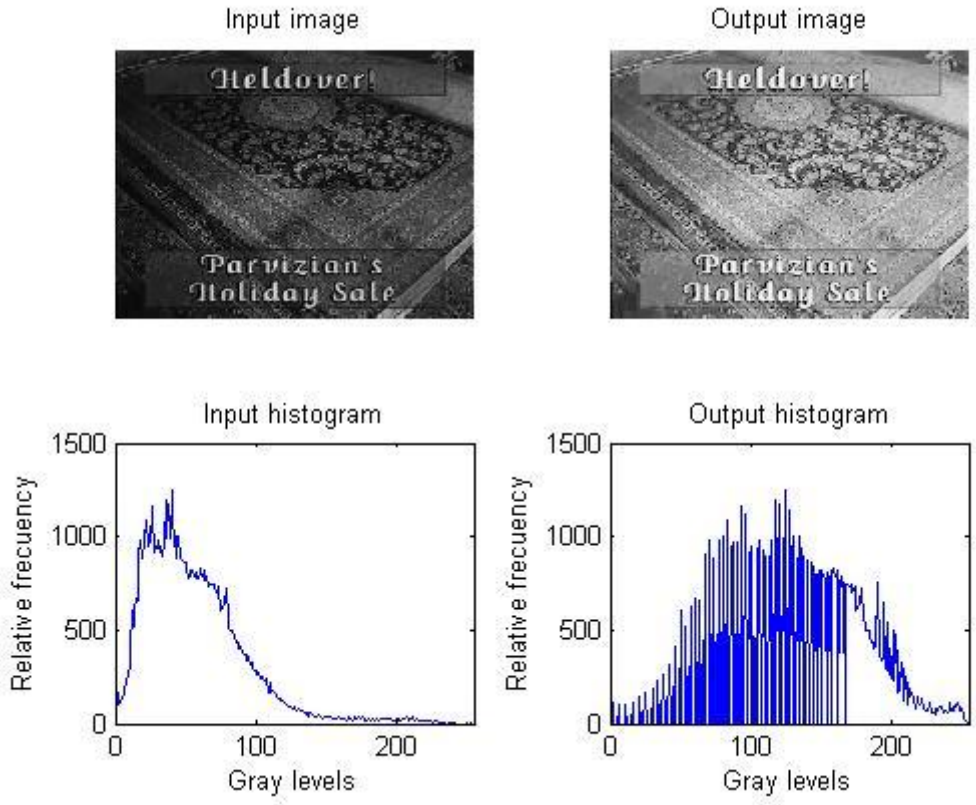


Figure 4.5.3 Contrast stretching, example 3.

The input image shows quite low contrast, the right side of the image it is the worse contrast seems like black. After a stretching with m value 60 and 2 of slope the image shows better contrast and the carpet is in all the image including the right side of the image. The first histogram is between 0 and 140 more or less and after the transformation the most values are between 50 and 200.

5. Conclusions

The subjective results depends on the input image, and each transformation works better for a type of image and worse for others types. So for dark images with low contrast the better results will be with the logarithm and the power law transformations using in the second one gamma values lower than 1. For light images it would be use the power law transformation with gamma higher than 1. For image with low contrast in grayscale the better methods are histogram equalization and contrast stretching. The second one works better if the middle gray levels have to be enhanced more than the black and white areas. Finally if the image has dark and light areas with low contrast and the objective is increase the contrast in that areas instead the all image it will be use the local enhancement.

Also it has to be measurement the human effort in the different algorithms. The only two that do not need human help are logarithm and histogram equalization. In power law transformation it has to be specified by the user the gamma value. In contrast stretching the m grayscale value and the slope of the transformation and local enhancement is the most hardworking function because it has to be calculate the limit values for the local deviation and local mean, and the size of the neighborhood.

Classify by time of execution: The algorithms implemented can be separate in three groups. The first one which average time is less than 0.1s, the second one between 0.1 and 0.4 seconds and the third with

more than 0.4 seconds. In the first group there be the logarithm transformation and the contrast stretching both of them are the fastest algorithms implemented in this thesis. The power-law transformation and the histogram equalization are in the second group. And in the last group it is the local enhancement method which is the slowest one because it have to make calculations for each block.

In few words, each function it is designed for a different type of image and works worse than other function in others images.

6. Abbreviations

FFT – Fast Fourier Transform

CRT – Cathode Ray Tube

TV – Television

7. Bibliography

1. Rafael C. Gonzalez y Richard E. Woods. Digital Image Processing. Second edition. Prentice Hall 2002.

2. Contrast limited adaptive histogram equalization.

   http://pdfdatabase.com/download/contrast-limited-adaptive-histogram-equalization-jodi-1998-pdf-7338978.html

3. Image enhancement for face recognition.

   http://handysolution.com/science/Image_Enhancement_for_Face_Recognition.pdf

4. Image processing Toolbox for use with MatLab. User guide.

5. John L. Semmlow. Biosignal and Biomedical Image processing.

6. Shape preserving contrast enhancement.

   http://www.iua.upf.es/~vcaselles/papers_v/LocalHistoConf.pdf

7. Transform coefficient histogram based image enhancement algorithms using contrast entropy.

   http://www.ece.tufts.edu/~karen/papers/Transform%20Coefficient%20Histogram%20Based%20Image%20Enhancement%20Algorithms%20using%20Contrast%20Entropy.pdf

8. Using levels to control contrast.

   http://www.luzette.com/download/Using%20levels%20to%20control%20contrast.pdf

9. Adaptative binarization.

   http://robotics.pme.duth.gr/pubs/Conferences/ADAPTIVE%20DOCUMENT%20BINARIZATION%20A%20human%20vision%20approach.pdf

10. Dah-Chung Chang and Wen-Rong Wu. Image contrast enhancement based on a histogram transformation of local standard deviation.

11. Bakul Vaghela, Shilpa Palnitkar, B Kartikeyan, and Santanu Chowdhury. Adaptive contrast enhancement technique for scanned documents, maps and remote sensing imagery.

12. J. Alex Stark. Adaptive image contrast enhancement using generalizations of histogram equalization.

13. Tarik Arici, Yucel Altunbasak. Image local contrast enhancement using adaptive non-linear filters.

IMAGE CONTRAST ENHANCEMENT METHODS

SUPERVISOR: Assoc. Prof. Antoaneta Popova

STUDENT: Cristian Ordoyo Casado

- The principal objective of enhancement is to process an image so that the result is more suitable than the original image for a specific application. Get a “better” image.

- Spatial domain methods and frequency domain methods.

- The term spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image.

$$g(x,y) = T[f(x,y)]$$

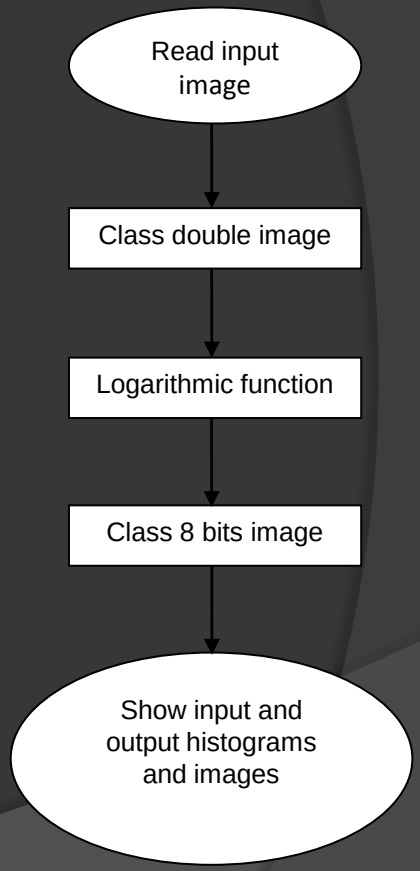
- where f(x, y) is the input image, g(x, y) is the processed image, and T is an operator on f, defined over some neighborhood of (x, y).

- Frequency domain processing techniques are based on modifying the Fourier transform of the image.
- Fast Fourier Transform (FFT) algorithm 1950s.
- More suitable for filtering spectrums.

$$s = c * \log(1 + r)$$

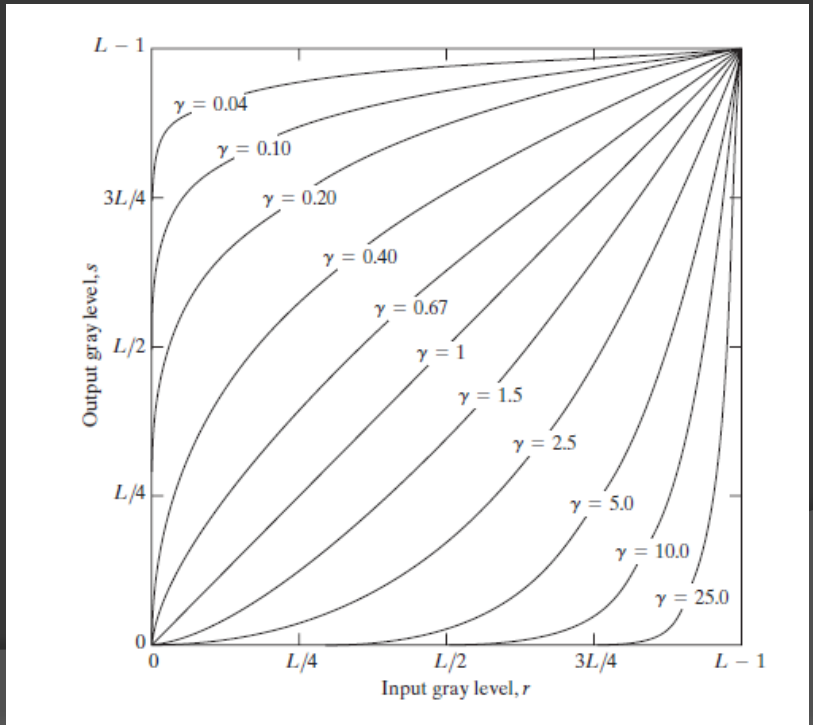
- where s is the output value, r is the input value and c is a constant.
- This transformation maps a narrow range of low gray-level values in the input image into a wider range of output levels.

```
input_image_process=double(input_image);


output_image_process=log(1+input_image_pr
    ocess);

output_image=im2uint8(mat2gray(output_imag
    e_process));

input_hist=imhist(input_image);
output_hist=imhist(output_image);
```

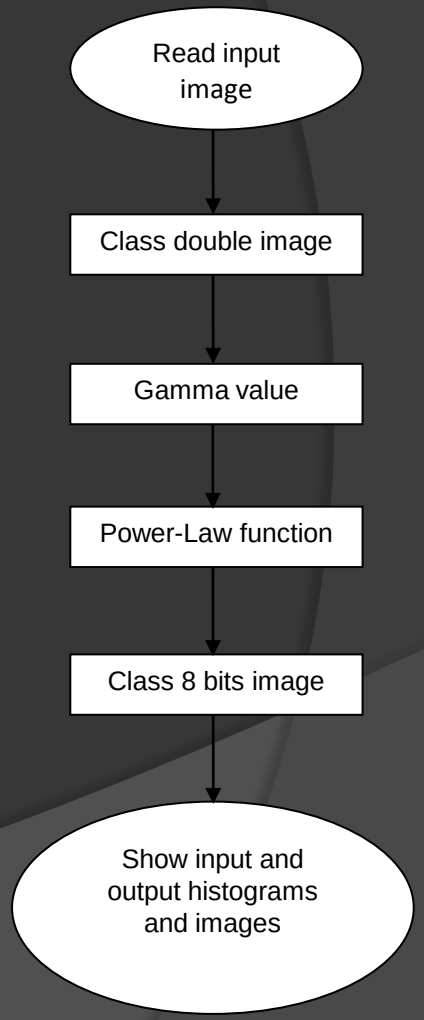


$$s = c * r^{\gamma}$$

- where c and γ are positive constants.

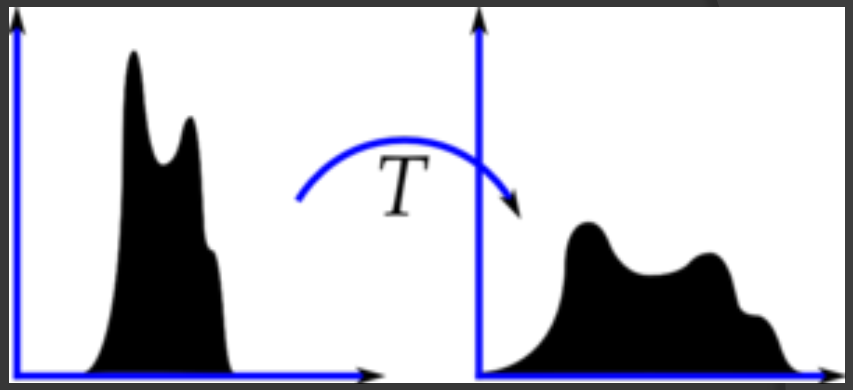


```
gamma=2;

output_image_process=imadjust(input_image
    _process,[0 1],[0 1],gamma);

output_image=im2uint8(mat2gray(output_ima
    ge_process));

input_hist=imhist(input_image);
output_hist=imhist(output_image);
```



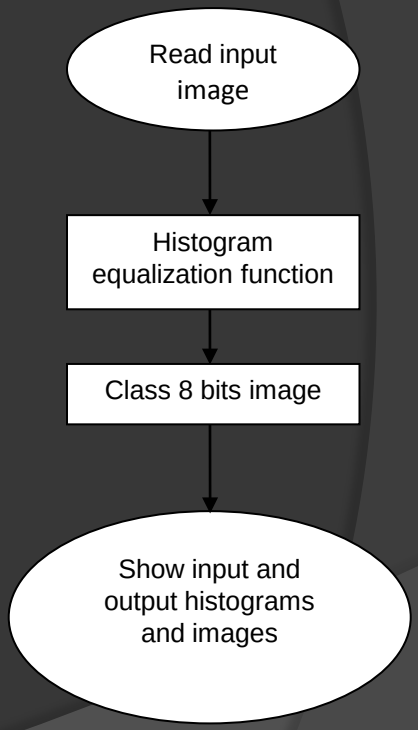
$$s_k = \frac{(L-1) * (r_k - r_{kmin})}{(r_{kmax} - r_{kmin})}$$

k = 0,1,2....L-1



- where r and s are the input and output pixels of the image, L is the different values that can be the pixels, and r_{kmax} and r_{kmin} are the maximum and minimum gray values of the input image.

```
output_image_process=histeq(input_ima
    ge_process);

output_image=im2uint8(mat2gray(output
    _image_process));

input_hist=imhist(input_image);
output_hist=imhist(output_image);
```

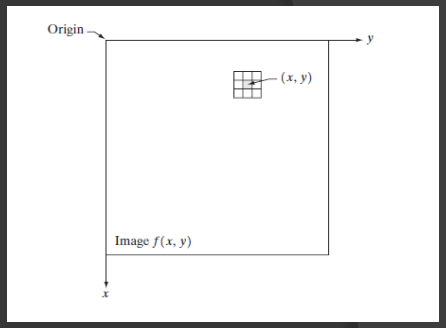


$$m = \sum_{i=0}^{L-1} r_i * p(r_i)$$

$$\sigma^2 = \sum_{i=0}^{L-1} (r_i - m)^2 * p(r_i)$$

$$m_{s_{xy}} = \sum_{(s,t) \in s_{xy}} r_{s,t} * p(r_{s,t})$$

$$\sigma^2_{s_{xy}} = \sum_{(s,t) \in s_{xy}} (r_{s,t} - m_{s_{xy}})^2 * p(r_{s,t})$$

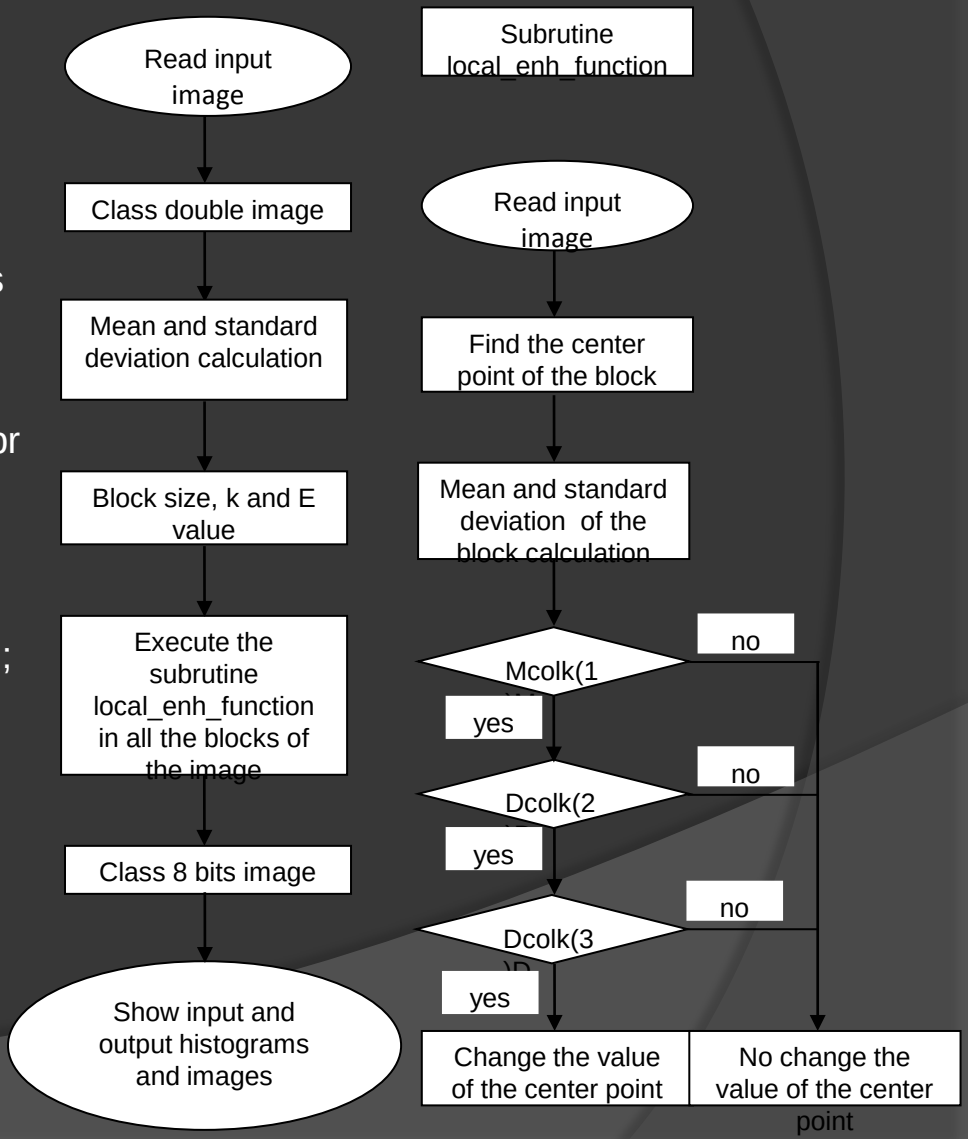


- Expresions for calculate the mean and the standard deviation in the whole image and in the neighborhood. Where $r_{s,t}$ is the gray level at coordinates (s, t) in the neighborhood, and $p(r_{s,t})$ is the neighborhood normalized histogram component corresponding to that value of gray level.

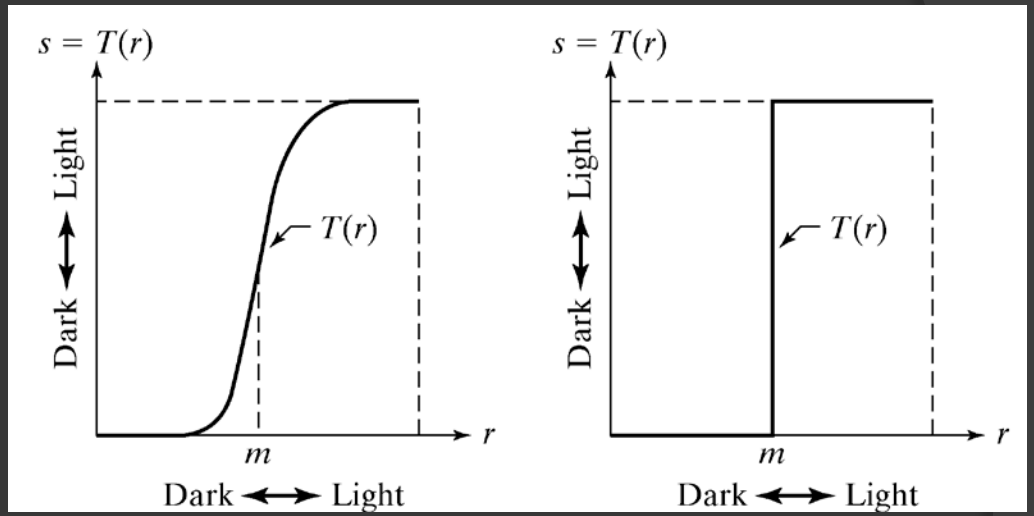
analysis
methods
experimental results
conclusions

logarithmic transformation
power-law transformation
histogram equalization
local enhancement using statistics
contrast stretching

```
M=mean2(input_image_process);
D=std2(input_image_process);

Bsize=[3 3]; k=[0.4 0.02 0.4]; E=4;

output_image_process=colfilt(input_image_proces
      s,Bsize,'sliding',
      @local_enh_function,M,D,E,k);

output_image=im2uint8(mat2gray(output_image_pr
      ocess));


function g=local_enh_function(Icol,M,D,E,k)
 Bcenter=floor((size(Icol,1)+1)/2); g=Icol(Bcenter,:);

Mcol=mean(Icol);
Dcol=std(Icol);

%Build the local response.
change=find((Mcol<=k(1)*M) & (Dcol>=k(2)*D) &
      (Dcol<=k(3)*D));
nochange=find((Mcol>k(1)*M) | (Dcol<k(2)*D) |
      (Dcol>k(3)*D));
g(change)=E*Icol(Bcenter,change);
```

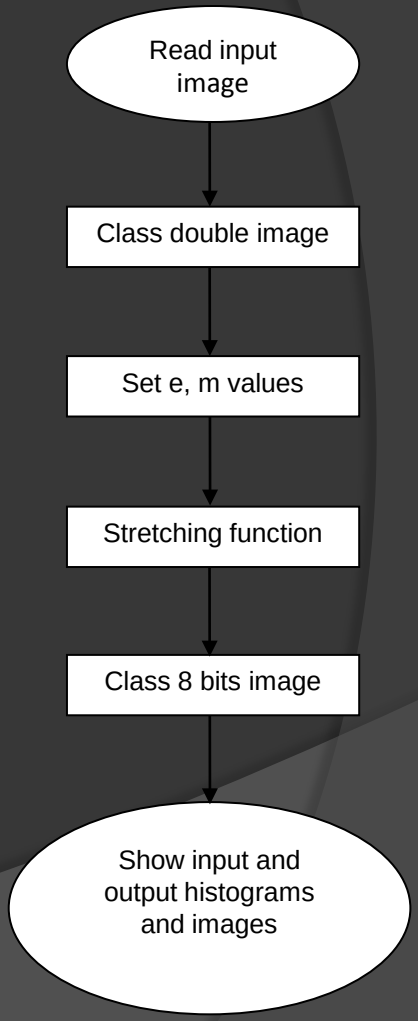


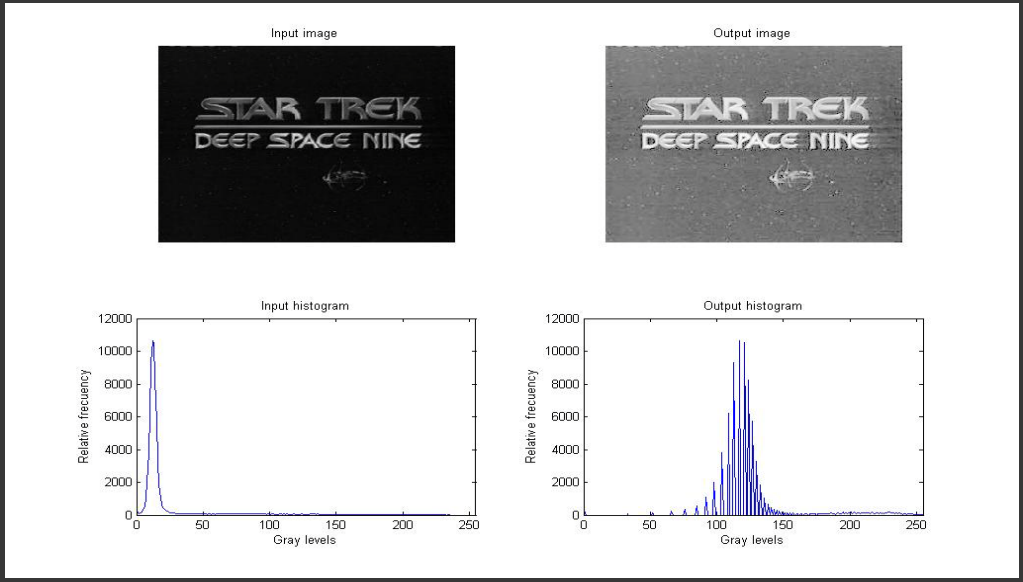
$$s = \frac{1}{1+(m/r)^E}$$



- where r are the input image values, s are the output image values, m is the thresholding value and E the slope.

```
input_image_process=double(input_image);

e=3;
m=80;

output_image_process = 1 ./ (1 +
    (m./input_image_process).^e);

output_image=im2uint8(mat2gray(output_ima
    ge_process));

input_hist=imhist(input_image);
output_hist=imhist(output_image);
```

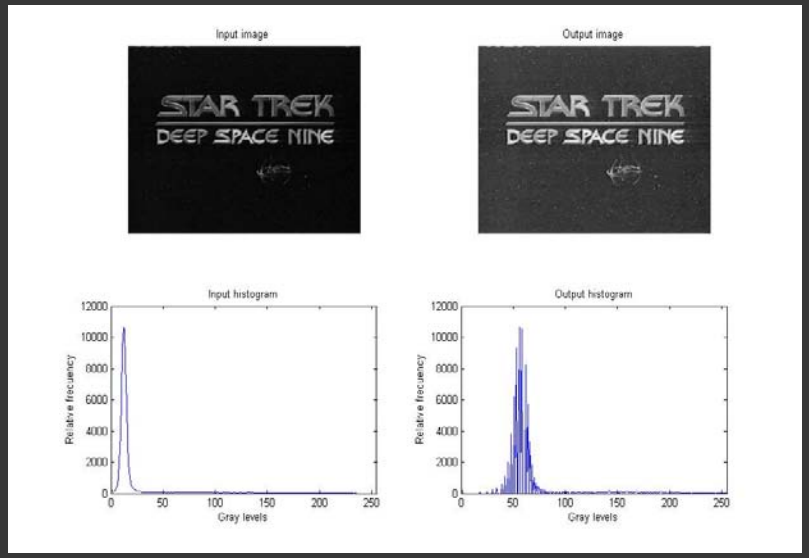


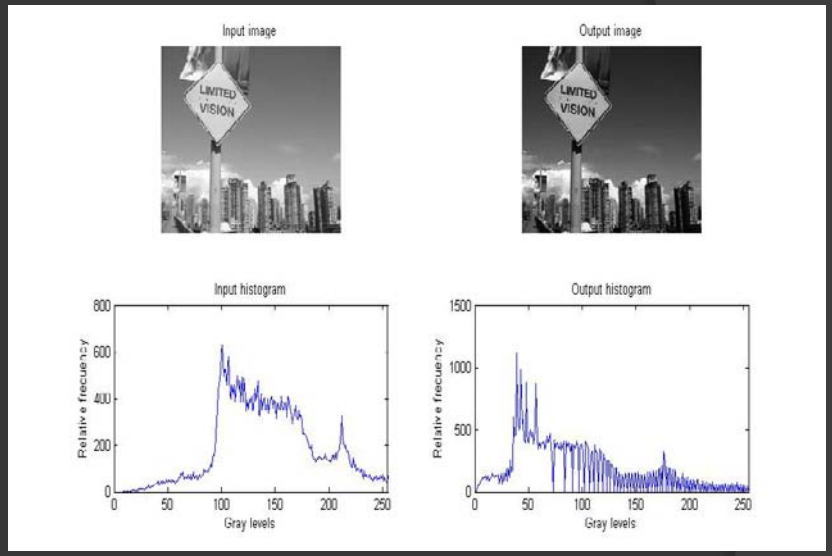


- Dark areas get more detail.
- Image became lighter.

analysis
methods
experimental results
conclusions

logarithmic transformation
power-law transformation
histogram equalization
local enhancement using statistics
contrast stretching

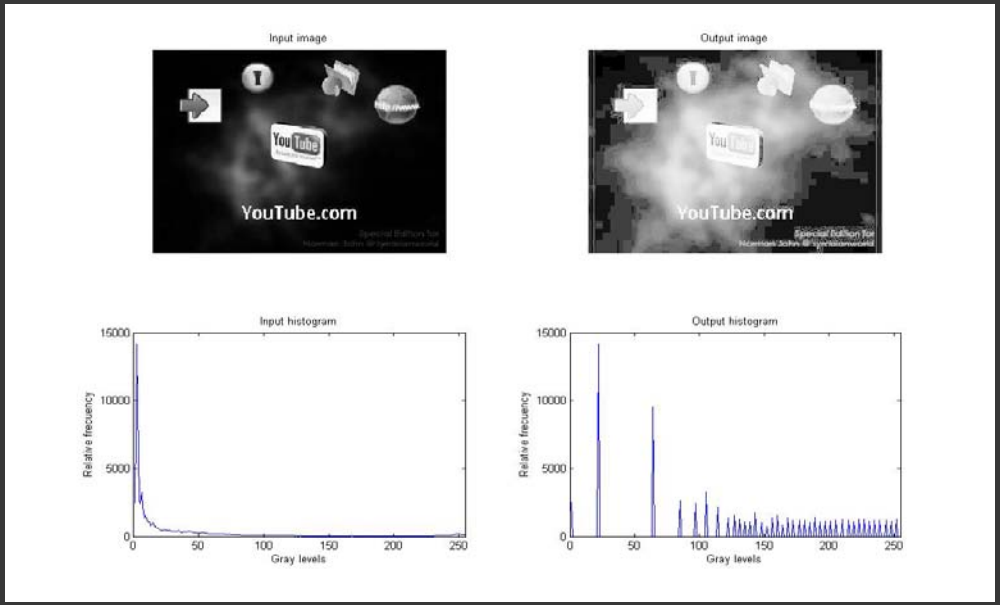




- Gamma = 0.5 < 1
- Dark areas get more detail.
- Image became lighter.

- Gamma = 2 > 1
- Light areas get more detail.
- Image became darker.

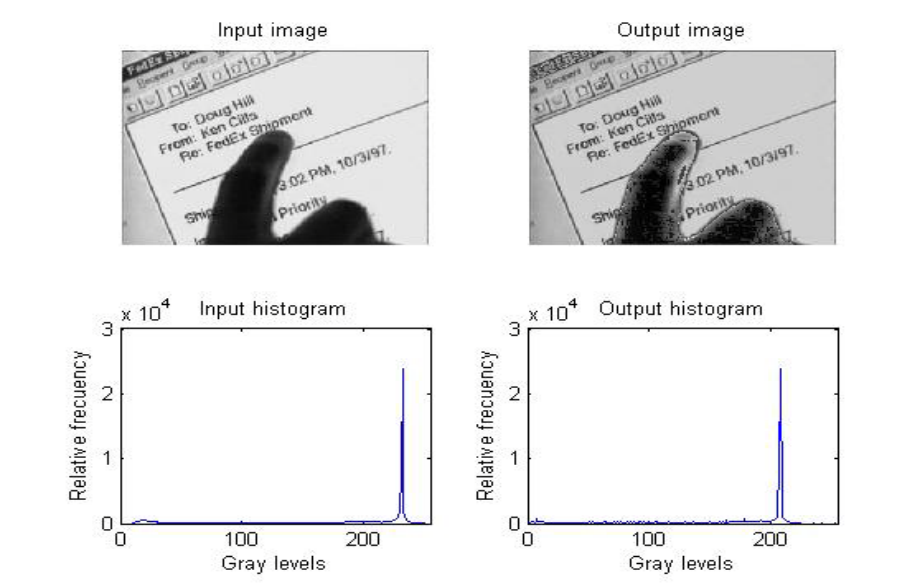
logarithmic transformation
power-law transformation
histogram equalization
local enhancement using statistics
contrast stretching



- It transform histograms with few different gray values into histograms with lots of different gray values. Giving more detail to close gray values.

analysis
methods
experimental results
conclusions

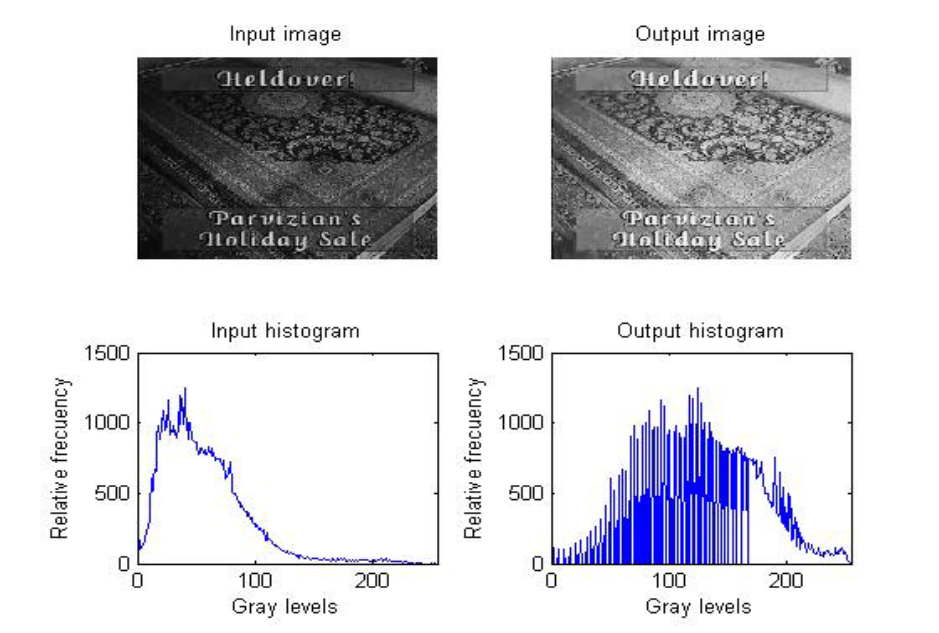
logarithmic transformation
power-law transformation
histogram equalization
local enhancement using stadistics
contrast stretching



- Gives more contrast to small areas that are no significatives in the whole image.

analysis
methods
experimental results
conclusions

logarithmic transformation
power-law transformation
histogram equalization
local enhancement using statistics
contrast stretching



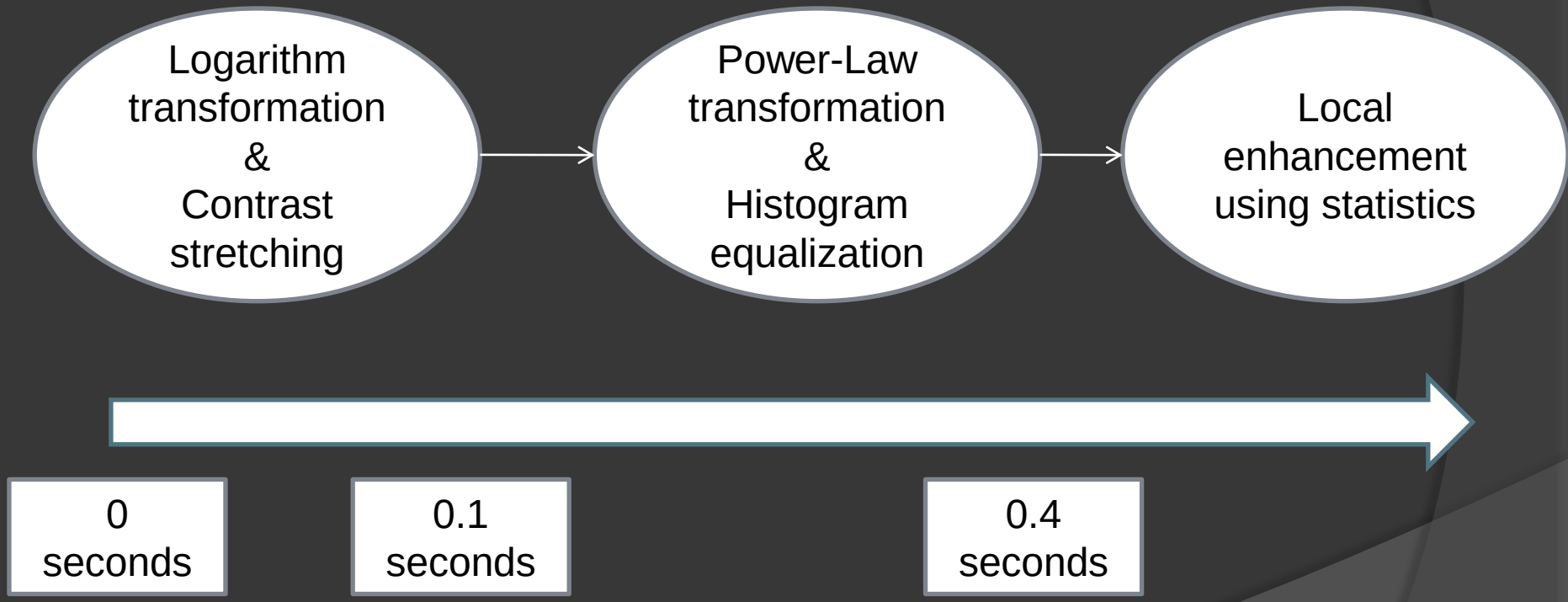
- The input images that have histograms in a short range of gray values became into histograms with more gray values giving more detail to the slope values of the contrast stretching function.

- For dark images with low contrast the better results will be with the logarithm and the power law transformations using in the second one gamma values lower than 1.
- For light images it would be use the power law transformation with gamma higher than 1.
- For image with low contrast in grayscale the better methods are histogram equalization and contrast stretching.
- Local enhancement if the image has dark and light areas with low contrast and the objective is increase the contrast in that areas instead the all image.

- Logarithm and histogram equalization NO need human intervention.
- In power law transformation it has to be specified by the user the gamma value.
- In contrast stretching the m gray level and the slope of the transformation.
- Local enhancement is the most hardworking function because it has to be calculate the limit values for the local deviation and local mean, and the size of the neighborhood.

analysis
methods
experimental results
conclusions

type of images
human intervention
time



Questions?